2021

First-order distributed optimization methods for machine learning with linear speed-up

BOSTON UNIVERSITY

COLLEGE OF ENGINEERING

Dissertation

FIRST-ORDER DISTRIBUTED OPTIMIZATION METHODS FOR MACHINE LEARNING WITH LINEAR SPEED-UP

by

ARTIN SPIRIDONOFF

B.Sc., Sharif University of Technology, 2016

Submitted in partial fulfillment of the

requirements for the degree of

Doctor of Philosophy

2021

Approved by

First Reader

Alexander Olshevsky, Ph.D.
Associate Professor of Electrical and Computer Engineering
Associate Professor of Systems Engineering

Second Reader

Ioannis Ch. Paschalidis, Ph.D.
Professor of Electrical and Computer Engineering
Professor of Systems Engineering
Professor of Biomedical Engineering
Professor of Computing & Data Sciences

Third Reader

Francesco Orabona, Ph.D.
Associate Professor of Electrical and Computer Engineering
Associate Professor of Systems Engineering
Associate Professor of Computing & Data Sciences
Associate Professor of Computer Science

Fourth Reader

Na Li, Ph.D.
Gordon McKay Professor of Electrical Engineering and Applied Mathematics
Harvard University

Acknowledgments

I would like to take this opportunity to thank my advisors, Professor Paschalidis and Professor Olshevsky, for their endless help and guidance, for teaching me how to become a better researcher, without whom this work wasn't possible. I would like to thank my thesis committee members Professor Orabona and Professor Li, for their feedback and careful work in reading this thesis, which was considerably improved as a result of their comments. I am thankful for the amazing Professors and teachers at Boston University, Professor Orabona, Professor Vakili, Professor Saenko, and Professor Cassandras, just to name a few, for teaching, inspiring, and sharing their intuition and wisdom with me and many other students. I would like to thank my colleagues, lab mates, and classmates, Ruidi Chen, Shi Pu, Qianqian Ma, Andrew Cutler, Arman Karimian, Arian Houshmand, and many more for brainstorming ideas with me and their inputs and guidance throughout my Ph.D. program. Most importantly, I would like to thank my family and friends, for their constant support, from thousands of miles away and in person, helping me get through the challenges of doing a Ph.D. abroad.

FIRST-ORDER DISTRIBUTED OPTIMIZATION METHODS FOR MACHINE LEARNING WITH LINEAR SPEED-UP

ARTIN SPIRIDONOFF

Boston University, College of Engineering, 2021

Major Professors: Alexander Olshevsky, Ph.D.
Associate Professor of Electrical and Computer Engineering
Associate Professor of Systems Engineering

Ioannis Ch. Paschalidis Ph.D.
Professor of Electrical and Computer Engineering
Professor of Biomedical Engineering
Professor of Systems Engineering
Professor of Computing and Data Sciences

ABSTRACT

This thesis considers the problem of average consensus, distributed centralized and decentralized Stochastic Gradient Descent (SGD) and their communication requirements. Namely, (i) an algorithm for achieving consensus among a collection of agents is studied and its convergence to the average is shown, in the presence of link failures and delays. The new results improve upon the prior works by relaxing some of the restrictive assumptions on communication, such as bounded link failures and intercommunication intervals, as well as allowing for message delays. Next, (ii) a Robust Asynchronous Stochastic Gradient Push (RASGP) algorithm is proposed to minimize the separable objective $F(\mathbf{z}) = \sum_{i=1}^{n} f_i(\mathbf{z})$ in a harsh network setting characterized by asynchronous updates, message losses and delays, and directed communication. RASGP is shown to asymptotically perform as well as the best bounds on a centralized gradient descent that takes steps in the direction of the sum of the noisy gradients of all local functions $f_i(\mathbf{z})$. Next, (iii) a new communication strategy for Local SGD is proposed, a centralized optimization algorithm where workers make local updates and then calculate their average values only once in a while. It is shown that linear speed-up in the number of workers N is possible, using only $\mathcal{O}(N)$ communication (averag-

ing) rounds, independent of the total number of iterations T. Empirical evidence suggests this bound is close to being tight as it is further shown that $\sqrt{N}$ or $N^{3/4}$ communications fail to achieve linear speed-up. Finally, (iv) under mild assumptions, the main of which is twice differentiability on any neighborhood of the optimal solution, one-shot averaging, which only uses a single round of communication, is shown to have optimal convergence rate asymptotically.

Contents

List of Tables

List of Figures

List of Abbreviations and Notations

OSA	.....	One-Shot Averaging
$\mathbb{R}$	.....	Set of Real numbers
RAPS	.....	Robust Asynchronous Push-sum
RASGP	.....	Robust Asynchronous Stochastic Gradient Push
SGD	.....	Stochastic Gradient Descent
SVM	.....	Support Vector Machine
$\|.\|$	.....	Euclidean norm of a vector or the spectral norm of a matrix.
$\|.\|_p$	.....	l_p-norm of a vector.
$\mathbf{0}$	.....	a vector of all zeros. Size is determined from the context.
$\mathbf{0}_n$	.....	$n \times 1$ column vector of all zeros.
$\mathbf{1}$	.....	a vector of all ones. Size is determined from the context.
$\mathbf{1}_n$	.....	$n \times 1$ column vector of all ones.
A_{ij}	.....	entry (i, j) of matrix $\mathbf{A}$.
$\mathcal{N}(\mu, \sigma^2)$	.....	Normal distribution with mean μ and variance σ^2.
$x_{\max}$	.....	$\max_i x_i$, unless mentioned otherwise.
$x_{\min}$	.....	$\min_i x_i$, unless mentioned otherwise.
$[z]$	.....	$\{1, \ldots, z\}$, for a positive integer z.

We use lower-case bold letters to represent vectors and upper-case bold letters to represent matrices.

A matrix is called *(row) stochastic* if it is non-negative and the sum of the elements of each row equals to one. Similarly, a matrix is *column stochastic* if its transpose is stochastic. A matrix is called *doubly stochastic* if it is both column and row stochastic.

Node i is an *in-neighbor* of node j, if there is a directed link from i to j. Hence j would be an *out-neighbor* of node i. We denote the set of in-neighbors and out-neighbors of node i at time k with $N_i^{-,k}$ and $N_i^{+,k}$, respectively. Moreover, we denote the number of in-neighbors and out-neighbors of node i at time k with $d_i^{-,k}$ and $d_i^{+,k}$, as its *in-degree* and *out-degree*, respectively. If the graph is fixed, we will simply drop the index k in the aforementioned notations.

Chapter 1

Introduction

Optimization is involved whenever there is a decision, trade-off, and a "cost." Earliest traces of optimization date back to 17th century with Fermat's calculus-based work to identify optima followed by Gauss and Newton proposing the first iterative methods to search for an optimum. *Linear Programming* emerged after World War II when Dantzig developed the Simplex Method in 1947, which was used to solve problems in resource allocation, scheduling, economics, and military planning among others. Since then, optimization algorithms have evolved remarkably.

In the past decade, with the lower cost of data storage and increasing computational power, Machine Learning (ML) methods have been applied to countless applications in healthcare, smart cities, autonomous vehicles, routing and transportation, natural language processing, and other domains. Machine Learning is closely tied to optimization through a *loss function* that is to be (usually) minimized in many supervised learning models (including linear regression, logistic regression, ridge regression, the LASSO, support vector machines, and their variants). With the rapid growth of data set sizes, it is crucial to develop fast algorithms that are able to process these large data sets efficiently.

While in many cases, optimization problems can be solved on a single processor, this might not be desired or possible in some applications. In a network comprised of multiple agents (e.g., data centers, sensors, vehicles, smartphones, or various IoT devices) engaged in data collection, it is sometimes impractical to collect all the information in one place. For instance, in healthcare domain, large amount of data sets are often stored in different locations/hospitals which can not be transferred to a central data repository due to privacy concerns [Brisimi et al., 2018]. In multiagent control problems, distributed methods

are widely explored for maneuvering and coordination of multiple robots [Cao et al., 2012, Peng et al., 2017]. In a wireless sensor network where each node collects random measurements from an unknown parameter x, decentralized estimation methods are used to simlply transmit the estimate of x to the end user, rather than the raw data, especially when the data is large (e.g., for image or video sensors) [Barrenetxea et al., 2008]. In Machine Learning, it might be desirable to partition the training data among multiple processors to exploit parallel computing resources. Consequently, distributed optimization techniques are currently being explored for potential use in a variety of estimation and learning problems over networks such as large-scale machine learning, healthcare, control, and sensor networks (e.g., coverage control, He et al. [2015]) due to their advantages over centralized systems, such as scalability, robustness to faults and privacy. Particularly, we are interested in distributed methods that are able to achieve linear speed-up in the number of processors (workers).

Communication is an inseparable part of distributed optimization and learning [Nedić et al., 2018]. However, reliable communication requires time and energy. Hence, it might not always be guaranteed or it might be too costly to achieve. Therefore, in this thesis, we seek to alleviate the communication requirements of distributed optimization methods without substantially sacrificing performance. Specifically, our contributions can be divided into 3 categories: (i) average consensus [Olshevsky et al., 2018], (ii) network independence of distributed optimization [Spiridonoff et al., 2020], (iii) Local SGD and one-shot averaging [Spiridonoff et al., 2021], which are described below.

1.1 Contributions of the thesis

1.1.1 Average consensus

When the agents in a network wish to agree on the same value, we say that they seek *consensus*. A simple strategy to reach consensus is by repeatedly updating each agent's variable to the average of its neighbors' values in the network [DeGroot, 1974, Tsitsiklis, 1984, Bertsekas and Tsitsiklis, 1989]. When the agents specifically want to find the average

of their initial values, we call the corresponding algorithm *average consensus*.

This simple problem formulation is useful in many distributed applications such as coverage control [Gao et al., 2008], distributed estimation and control [Garin and Schenato, 2010], distributed optimization [Tsianos et al., 2012b, Varagnolo et al., 2016] and many more. For example Bof et al. [2017a] use a robust version of push-sum (an algorithm that reaches average consensus over directed networks) as a building block to develop an asynchronous Newton-based distributed optimization algorithm that is robust to packet losses. See Olshevsky [2010, 2014] for a more in-depth discussion of consensus protocols and their applications.

Push-sum is one of the many algorithms for average consensus that was first proposed by Kempe et al. [2003]. This algorithm has been widely used to develop protocols that reach average consensus, under different assumptions and scenarios; such as the presence of bounded delays [Hadjicostis and Charalambous, 2014], time-varying graphs [Hadjicostis and Charalambous, 2012, Rezaeinia et al., 2017], or asynchronous communication [Bénézit et al., 2010]. Thus, we start this thesis by looking into consensus methods and their communication requirements.

Since reliable communication is a very restrictive assumption in network applications, or expensive to enforce, recent work has considered algorithms that reach consensus in a setting where communication between agents is unreliable. While in this case, push-sum might not converge to average, exponential convergence still holds and the error between the final value and the true average can be characterized Gerencsér and Hendrickx [2015].

Hadjicostis et al. [2016] introduced the technique of running sums (counters) and modified push-sum to overcome possible packet drops in a synchronous communication setting. Bof et al. [2017b] took this further and allowed for bounded asynchrony where only one agent makes an update at any time. These limitations motivated us to study and explore sufficient connectivity conditions which allow for any subset of agents to make updates and intercommunication intervals to be potentially unbounded. Our contributions in this part of the thesis can be summarized as follows:

- We propose a Robust Asynchronous Push-Sum (RAPS) algorithm for average consensus over a directed graph, which is both fully asynchronous and robust to unreliable communications. We show its convergence to the average value while allowing for slowly growing but potentially unbounded communication intervals.

- We show exponential convergence to the average for RAPS, in a harsh network setting, allowing for bounded delays in addition to bounded link failures and agents' update intermissions.

- We further provide convergence guarantees for RAPS when agents' iterates are perturbed every time they make an update.

1.1.2 Network independence of distributed optimization

Next, we consider the standard model of distributed optimization for the sum of functions $F(\mathbf{z}) = \sum_{i=1}^{n} f_i(\mathbf{z})$, where node i in a network holds the function $f_i(\mathbf{z})$. This separable model was first formally analyzed in Nedić and Ozdaglar [2009], where performance guarantees on a fixed step-size subgradient method were obtained. This fairly simple problem formulation is capable of capturing a variety of scenarios in estimation and learning. Informally, $\mathbf{z}$ is often taken to parameterize a model, and $f_i(\mathbf{z})$ is a loss function measuring how well $\mathbf{z}$ matches the data held by agent i. Agreeing on a minimizer of $F(\mathbf{z})$ means agreeing on a model that best explains all the data throughout the network.

Research on models of distributed optimization dates back to the 1980s, see Tsitsiklis et al. [1986]. The algorithms we will study here are based on the standard "consensus+gradient descent" framework: nodes will take steps in the direction of their gradients and then "reconcile" these steps by moving in the directions of an average of their neighbors in the graph. We refer the reader to Nedić et al. [2018], Yuan et al. [2016], for a more recent and simplified analysis of such methods. In what follows, we discuss the key features we consider.

Directed graphs: In some distributed applications, the network connectivity will be dependent on the proximity of nodes and their transmition power (e.g. wireless sensor networks). In such cases, the network topology is a directed graph. The study of distributed separable optimization over directed graphs was initiated in Tsianos et al. [2012b], where a distributed approach based on dual averaging with convex functions over a fixed graph was proposed and shown to converge at an $\mathcal{O}(1/\sqrt{k})$ rate.

The reason directed graphs present a problem is because much of distributed optimization relies on the primitive of "multiplication by a doubly stochastic matrix:" given that each node of a network holds a number x_i, the network needs to compute y_i, where $\mathbf{x} = (x_1, \ldots, x_n)^\top$, $\mathbf{y} = (y_1, \ldots, y_n)^\top$ and $\mathbf{y} = \mathbf{W}\mathbf{x}$ for some doubly stochastic matrix $\mathbf{W}$ with positive spectral gap. This is pretty easy to accomplish over undirected graphs (see Nedić et al. [2018]) but not immediate over directed graphs.

Asynchrony: It has been noted that asynchronous algorithms are often preferred to synchronous ones, due to the difficulty of perfectly coordinating all the agents in the network, e.g., due to clock drift. Therefore, a number of papers have studied asynchronicity in the context of distributed optimization. As an example, Agarwal and Duchi [2011] analyze the convergence of gradient-based optimization algorithms whose updates depend on delayed stochastic gradient information due to asynchrony.

Message losses: Dealing with message losses has always been a challenging problem for multi-agent optimization protocols. Recently, Hadjicostis et al. [2016] resolved this issue rather elegantly for the problem of distributed average computation by having nodes exchange certain running sums. It was shown in Hadjicostis et al. [2016] that the introduction of these running sums is equivalent to a lossless algorithm on a slightly modified graph. We will use the same approach in this work to deal with message losses.

Stochastic gradients: In many applications, calculating the exact gradients can be computationally very expensive or impossible [Lan et al., 2020]. In one possible scenario, nodes

are sensors that collect measurements at every step, which naturally corrupts all the data with noise. Alternatively, communication between agents may insert noise into the information transmitted between them. Finally, when $f_i(\mathbf{z})$ measures the fit of a model parameterized by the vector $\mathbf{z}$ to the data of agent i, it may be efficient for agent i to randomly select a subset of its data and compute an estimate of the gradient-based on only those data points [Alpcan and Bauckhage, 2009].

Distributed vs. centralized: Traditionally, the bounds derived on distributed methods were considerably worse than those derived for centralized methods. The breakthrough papers by Chen and Sayed [2015], Pu and Garcia [2017], Morral et al. [2017], were the first to address this gap. In Morral et al. [2017], it was proved for the first time, that distributed gradient descent with an appropriately chosen step-size, asymptotically performs similarly to a centralized method that takes steps in the direction of the sum of the noisy gradients (assuming iterates will remain bounded almost surely). Both Pu and Garcia [2017] and Morral et al. [2017] were over fixed, undirected graphs with no message loss or delays or asynchronicity.

One of our main concerns in this chapter is to develop decentralized optimization methods which perform as well as their centralized counterparts. Specifically, we will compare the performance of a distributed method for (3.1) on a network of n nodes with the performance of a centralized method which, at every step, can query all n gradients of the functions $f_1(\mathbf{z}), \ldots, f_n(\mathbf{z})$. Since the distributed algorithm gets noise-corrupted gradients, so should the centralized method. Thus, the natural approach is to compare the distributed method to centralized gradient descent which moves in the direction of the sum of the gradients of $f_1(\mathbf{z}), \ldots, f_n(\mathbf{z})$. This method of comparison keeps the "computational power" of the two nodes identical.

Our contributions to derive bounds on distributed methods comparable to a centralized one can be listed below:

- We allow for a harsh network model characterized by asynchronous updates, message

delays, unpredictable message losses, and directed communication among nodes.

- In this setting, we analyze a modification of the Gradient-Push method for distributed optimization, assuming that (i) node i is capable of generating gradients of its function $f_i(\mathbf{z})$ corrupted by zero-mean bounded-support additive noise at each step, (ii) $F(\mathbf{z})$ is strongly convex, and (iii) each $f_i(\mathbf{z})$ has Lipschitz gradients.

- We show that our proposed method asymptotically performs as well as the best bounds on centralized gradient descent that takes steps in the direction of the sum of the noisy gradients of all the functions $f_1(\mathbf{z}), \ldots, f_n(\mathbf{z})$ at each step.

1.1.3 Local SGD

We look into speeding up stochastic gradient descent (SGD) by parallelizing it across multiple workers. We assume the *same* data set is shared among N workers, who can take SGD steps and coordinate with a central server. The assumption is valid when the same data set is either shared across multiple workers in the same cluster, or the assignment of data points to workers is random so that any distributional differences are small. Sharing the data set across multiple workers in this way is a popular strategy to speed up training. For example, such data sharing is implemented in Chen et al. [2012], Yadan et al. [2013], Zhang et al. [2013a] to speed up the training of deep neural networks with multiple GPUs within a single sever. Unfortunately, this could require a lot of communication between the workers and the server, which can dramatically reduce the gains from parallelism. Therefore, *Local SGD* (also known as FedAvg) has been proposed to reduce communications [McMahan et al., 2017, Dieuleveut and Patel, 2019]. In this method, workers compute (stochastic) gradients and update their parameters locally, and communicate only once in a while to obtain the average of their parameters. While the initial analysis of Local SGD for general strongly convex functions [Stich, 2019] showed it needs $\Omega(\sqrt{T})$ communications for T total gradient steps in order for the error to scale proportionately to $1/(NT)$, this has been successively improved in a string of papers, with the state of the art requiring $\Omega\left(N\left(\text{ polynomial in } \log\left(T\right)\right)\right)$ communications [Stich and Karimireddy, 2019], [Khaled

et al., 2020]. Our contributions to Local SGD can be summarized as follows:

- We present a new analysis of Local SGD which allows for a general noise model for stochastic gradients as well as any arbitrary choice of communication times, whereas prior works simply assume a general upper bound H on the number of iterations between any two consecutive communication rounds.
- We present a new communication strategy with only a fixed number of communications independent of T: specifically, only $\Omega(N)$ communications are required.
- We show linear speed-up in the number of workers N is achieved using the proposed communication strategy.

1.1.4 One-shot averaging

Finally, we study an extreme case of Local SGD, in which workers make local steps until the very end and then they average their parameters, using only $R = 1$ communication round. The previous literature has shown OSA achieves asymptotic linear speed-up under some restrictive assumptions. For instance, Dieuleveut and Patel [2019] shows this for three times continuously differentiable functions with second and third derivatives uniformly bounded. Similarly, Godichon-Baggioni and Saadane [2020] requires the objective function to be strongly convex, twice continuously differentiable almost everywhere, with a bounded Hessian everywhere. Our contributions in OSA can be summarized as below:

- We show asymptotic linear speed-up in the number of workers, requiring mild assumptions, the main of which is twice differentiability on any neighborhood of the optimal solution.
- In numerical experiments, we show that twice differentiability at the optimum is necessary.

1.2 Future directions

There are many interesting directions to continue the work of this thesis, especially in the area of Federated Learning which has many challenges yet to be solved and to develop more practical and scalable methods. For a more detailed discussion, see Section 5.2

Chapter 2

Robust asynchronous consensus

2.1 Introduction

Consider a set of agents, whose goal is to reach consensus by exchanging information locally with their neighbors through a directed graph. There is a large body of work on consensus algorithms. Ordinary consensus has been shown to converge asymptotically under various scenarios such as growing intercommunicating intervals [Lorenz, 2011], presence of delays and/or unbounded intercommunication intervals [Blondel et al., 2005]. Another problem of interest for which extensive research has been carried out is *average* consensus. While most related works study asymptotic convergence, Charalambous et al. [2015] studies average consensus in a finite number of steps.

Reliable communication between the agents is not always guaranteed. Thus, Hadjicostis et al. [2016] introduced the technique of running sums (counters) and modified push-sum to overcome possible packet drops and imprecise knowledge of the network in a synchronous communication setting. They proved *almost surely* convergence of their algorithms using weak ergodicity. Inspired by Hadjicostis et al. [2016], Bof et al. [2017b] took this further and developed an asynchronous algorithm for average consensus, which is robust to unreliable communication. This algorithm uses a *broadcast asymmetric* communication protocol; that is, at each iteration only one node is allowed to wake up and transmit information to its neighbors. Exponential convergence of this algorithm is proved under bounded consecutive link failures and nodes' update delays.

Distributed *synchronous* systems require coordination between the agents. Asynchronous systems, in contrast, do not depend on global clock signals. This can save power as agents do not have to perform computation and communication at every iteration. How-

ever, it might require more iterations to converge. While existing works on push-sum in the presence of link failures assume synchronous [Hadjicostis et al., 2016] or broadcast asymmetric [Bof et al., 2017b] communication setting, our first contribution in this chapter is to develop a *fully asynchronous* robust push-sum algorithm that allows the successive link failures to grow to infinity. We additionally show the same algorithm converges exponentially to average under the assumption of bounded delays and link failure.

The rest of the chapter is organized as follows. In Section 2.2 we introduce our notation and define the problem. In Sections 2.3 and 2.4 we study ordinary consensus and push-sum algorithms, respectively, and state our convergence results. In Section 2.5, we propose an Robust Asynchronous Push-Sum (RAPS) algorithm and prove convergence guarantees under various scenarios, followed by concluding remarks in Section 2.6.

2.2 Problem formulation

2.2.1 Notations and definitions

To a non-negative matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ we associate a directed graph $\mathcal{G}_{\mathbf{A}}$ with vertex set $\mathcal{N} = \{1, 2, \ldots, n\}$ and edge set $\mathcal{E}_{\mathbf{A}} = \{(i, j)|A_{ji} > 0\}$. Note that the graph might contain self-loops. By $[\mathbf{A}]_\alpha$ we denote the *thresholded* matrix obtained by setting every element of $\mathbf{A}$ smaller than α to zero. Given a sequence of matrices $\mathbf{A}^0, \mathbf{A}^1, \mathbf{A}^2, \ldots$, we denote by $\mathbf{A}^{k_2:k_1}, k_2 \geq k_1$, the product of elements k_1 to k_2 of the sequence, inclusive, in the following order:

$$\mathbf{A}^{k_2:k_1} = \mathbf{A}^{k_2}\mathbf{A}^{k_2-1}\cdots\mathbf{A}^{k_1}.$$

We sometimes use the notion of *mass* to denote the value an agent holds, sends or receives. With that in mind, we can think of a value being sent from one node, as a mass being transferred.

2.2.2 Problem formulation

Consider a set of n agents $\mathcal{N} = \{1, 2, \ldots, n\}$, where each agent i holds an initial scalar value $x_i(0)$. These agents communicate with each other through a sequence of directed graphs.

Our goal is to develop protocols through which these agents communicate and update their values so that they reach consensus. Throughout this chapter we use the terms *agents* and *nodes* interchangeably.

Ordinary consensus and push-sum are two main algorithms proposed for this purpose. In ordinary consensus, each node updates its value by forming a convex combination of the values of its in-neighbors. In push-sum, average consensus is reached by running two parallel iterations in which, each node splits and sends its value to its out-neighbors and updates its own value by forming the sum of the messages that it has received.

2.3 Ordinary consensus

Although the main target of this chapter is push-sum, in this section we state and prove similar results for ordinary consensus. Comparable results can be found in Lorenz [2011], however the proofs provided here are necessary to understand the methods used in the following sections.

Linear consensus is defined as,

$$\mathbf{x}(k+1) = \mathbf{A}^k \mathbf{x}(k), \quad k = 0, 1, \ldots, \tag{2.1}$$

where the matrices $\mathbf{A}^k$ are stochastic and $\mathbf{x}(k)$ is constructed by collecting all $x_i(k)$ in a column vector. Under the following conditions, the iteration (2.1) results in consensus, meaning all the $x_i(k)$ converge to the same value as $k \to \infty$.

The following assumption ensures sufficient connectivity of the graphs.

Assumption 1. *There exist a sequence $b_1, b_2, \ldots$ of positive integers such that when we partition the sequence of graphs $\mathcal{G}^0, \mathcal{G}^1, \mathcal{G}^2, \ldots$ to consecutive blocks of length b_k, $k = 1, 2, \ldots$, the graph constructed by the union of the edges in each block, is strongly connected. Also each graph $\mathcal{G}^k$ has a self-loop at every node.*

Let us define $\mu_0 = \lambda_0 = 0$, and for $k \geq 1$:

$$\mu_k = \sum_{j=1}^{k} b_j, \tag{2.2}$$

$$\lambda_k = \sum_{j=(k-1)n+1}^{kn} b_j = \mu_{kn} - \mu_{(k-1)n}. \tag{2.3}$$

The following proposition states sufficient conditions for the convergence of ordinary consensus with growing intercommunication intervals.

Proposition 1. *Suppose there exist some $\alpha > 0$ such that the sequence of graphs $\mathcal{G}_{[\boldsymbol{A}^0]_\alpha}$, $\mathcal{G}_{[\boldsymbol{A}^1]_\alpha}$, $\mathcal{G}_{[\boldsymbol{A}^2]_\alpha}, \ldots$ satisfies Assumption 1. If there exist some $K \geq 1, T \geq 0$, such that $\lambda_k \leq -\frac{\ln(k+T)}{\ln(\alpha)}$ for all $k \geq K$, then $\mathbf{x}(k)$ converges to a limit in span$\{\boldsymbol{1}\}$.*

Before proving the proposition, we need the following lemmas.

Lemma 1. *Suppose there exists some $\alpha > 0$ such that the sequence of graphs $\mathcal{G}_{[\boldsymbol{A}^0]_\alpha}, \mathcal{G}_{[\boldsymbol{A}^1]_\alpha}$, $\mathcal{G}_{[\boldsymbol{A}^2]_\alpha}, \ldots$ satisfies Assumption 1. Then for $l \geq 0$, $\boldsymbol{A}^{\mu_{l+n}-1:\mu_l}$ is a strictly positive matrix, with its elements at least $\alpha^{\mu_{l+n}-\mu_l}$.*

Proof. Consider the set of reachable nodes from node i in time period k_1 to k_2 in the graph sequence $\mathcal{G}_{[\mathbf{A}^0]_\alpha}, \mathcal{G}_{[\mathbf{A}^1]_\alpha}, \mathcal{G}_{[\mathbf{A}^2]_\alpha}, \ldots$, and denote it by $N^{k_2:k_1}$. Since by Assumption 1 each of these graphs has self-loop at every node, the set of reachable nodes never decreases. If $N^{\mu_{l+m}-1:\mu_l} \neq \{1, 2, \ldots, n\}$ then $N^{\mu_{l+m+1}-1:\mu_l}$ is a strict super-set of $N^{\mu_{l+m}-1:\mu_l}$; because in period μ_{l+m} to $\mu_{l+m+1} - 1$ there is an edge in some $\mathcal{G}_{[\mathbf{A}^i]_\alpha}$ leading from the set of reachable nodes from i, to those not reachable from i; this is true because the union of the graphs in block μ_{l+m} to $\mu_{l+m+1} - 1$ is strongly connected. Hence we conclude $N^{\mu_{l+n}-1:\mu_l} = \{1, 2, \ldots, n\}$ and $\mathbf{A}^{\mu_{l+n}-1:\mu_l}$ is strictly positive. Furthermore, since every positive element of $[\mathbf{A}^k]_\alpha$ is at least α by construction, every element of $\mathbf{A}^{\mu_{l+n}-1:\mu_l}$ is at least $\alpha^{\mu_{l+n}-\mu_l}$. □

Lemma 2. *Suppose $\boldsymbol{A}$ is a stochastic matrix with entries at least $\beta > 0$. If $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{u}$ then,*

$$v_{\max} - v_{\min} \leq (1 - n\beta)\left(u_{\max} - u_{\min}\right). \tag{2.4}$$

This lemma is proved in Seneta [2006, Theorem 3.1 & Exercise 3.8].

Lemma 3. *Suppose $\boldsymbol{A}$ is a stochastic matrix and $\boldsymbol{v} = \boldsymbol{A}\boldsymbol{u}$. Then for all i,*

$$u_{\min} \leq v_i \leq u_{\max}. \tag{2.5}$$

This lemma holds true because each v_i is a convex combination of elements of $\mathbf{u}$.

Lemma 4. *Suppose* $0 < \alpha_k < 1$ *for* $k = 1, \ldots, \infty$, *then* $\prod_{k=1}^{\infty}(1-\alpha_k) = 0$ *if and only if* $\sum_{k=1}^{\infty}\alpha_k = \infty$.

This lemma is proved in Brémaud [2013, Appendix: Theorem 1.9] and we will skip the proof here.

Proof of Proposition 1. By Lemma 1, we have for $k \geq 1$,

$$\left[\mathbf{A}^{\mu_{kn}-1:\mu_{(k-1)n}}\right]_{ij} \geq \alpha^{\mu_{kn}-\mu_{(k-1)n}} = \alpha^{\lambda_k}.$$

Applying Lemma 2, we get,

$$x_{\max}(\mu_{kn}) - x_{\min}(\mu_{kn}) \leq \left(1 - n\alpha^{\lambda_k}\right)\left(x_{\max}(\mu_{(k-1)n}) - x_{\min}(\mu_{(k-1)n})\right). \tag{2.6}$$

Hence, using (2.6) for $k = 1, \ldots, l$ we obtain,

$$x_{\max}(\mu_{ln}) - x_{\min}(\mu_{ln}) \leq \prod_{k=1}^{l}\left(1 - n\alpha^{\lambda_k}\right)\left(x_{\max}(0) - x_{\min}(0)\right).$$

We have $0 < \alpha < 1$ and $\lambda_k \leq -\frac{\ln(k+T)}{\ln(\alpha)}$ for all $k \geq K$. It follows,

$$\begin{aligned}\sum_{k=1}^{\infty} n\alpha^{\lambda_k} &\geq \sum_{k=K}^{\infty} n\alpha^{\lambda_k} \geq \sum_{k=K}^{\infty} n\alpha^{-\frac{\ln(k+T)}{\ln(\alpha)}} \\ &= \sum_{k=K}^{\infty} n\left(\alpha^{\frac{1}{\ln(\alpha)}}\right)^{-\ln(k+T)} = \sum_{k=K}^{\infty}\frac{n}{k+T} = \infty.\end{aligned}$$

Using Lemmas 3 and 4 and (2.6) we conclude that Proposition 1 holds. □

Proposition 1 proves the convergence of $x_i(k)$'s to a value which is not necessarily the total average and depends on the sequence of matrices. However if the matrices $\mathbf{A}^k$ are doubly stochastic, the sum of the values of all nodes (agents) is preserved and therefore the algorithm converges to *average* consensus.

2.4 Push-sum

Push-sum is an algorithm that reaches average consensus and does not require *doubly* stochastic matrices, as opposed to ordinary average consensus. The central idea of the

classic push-sum method [Kempe et al., 2003] to deal with directed communication, is to have a separate update equation for the y-variables, which informs us how we should rescale the x-variables. Here, we assume each node knows its out-degree at every iteration. Under this assumption, it turns out that average consensus is possible and may be accomplished using the following iteration,

$$\begin{aligned} x_i(k+1) &= \sum_{j \in N_i^{-,k}} \frac{x_j(k)}{d_j^{+,k}}, \\ y_i(k+1) &= \sum_{j \in N_i^{-,k}} \frac{y_j(k)}{d_j^{+,k}}, \\ z_i(k+1) &= \frac{x_i(k+1)}{y_i(k+1)}, \end{aligned} \tag{2.7}$$

where the auxiliary variables y_i are initialized as $y_i^0 = 1$ and are collected in a column vector $\mathbf{y}$. This iteration is implemented in a distributed way using two steps. First each node i broadcasts $x_i(k)/d_i^{+,k}$ to its out-neighbors. Next, every node sets $x_i(k+1)$ to be the sum of the incoming messages. Variables $y_i(k)$ follow the same evolution. $z_i(k)$ will be node i's estimation of the average.

We define $\mathbf{W}^k$ to be the matrix such that iteration (2.7) may be written as,

$$\begin{aligned} \mathbf{x}(k+1) &= \mathbf{W}^k \mathbf{x}(k), \\ \mathbf{y}(k+1) &= \mathbf{W}^k \mathbf{y}(k). \end{aligned}$$

Next, we will state a proposition regarding the sufficient conditions for the push-sum algorithm to converge.

Proposition 2. *Suppose the sequence of graphs $\mathcal{G}_{\boldsymbol{W}^0}, \mathcal{G}_{\boldsymbol{W}^1}, \mathcal{G}_{\boldsymbol{W}^2}, \ldots$, satisfies Assumption 1. If there exist some $K \geq 1, T \geq 0$, such that $\lambda_k \leq \frac{\ln(k+T)}{2\ln(n)}$ for all $k \geq K$, by implementing the push-sum algorithm (2.7), it follows*

$$\lim_{k \to \infty} z_i(k) = \frac{\sum_{j=1}^n x_j(0)}{n}.$$

Note that positive elements of $\mathbf{W}^k$ are at least $1/d_{\max}^{+,k} \geq 1/n$. Moreover, $\mathbf{W}^k$ is column

stochastic, i.e.,

$$\mathbf{1}^T\mathbf{W}^k = \mathbf{1}^T.$$

consequently, the sum of $x(k)$ and $y(k)$ are preserved, i.e., for $k \geq 0$,

$$\sum_{i=1}^{n} x_i(k) = \sum_{i=1}^{n} x_i(0), \tag{2.8}$$

$$\sum_{i=1}^{n} y_i(k) = \sum_{i=1}^{n} y_i(0) = n. \tag{2.9}$$

Before proving the proposition, we need the following lemma, which establishes bounds for $y_i(\mu_{ln})$, $l \geq 1$.

Lemma 5. *Suppose the Assumptions stated in Proposition 2 are satisfied. The following bounds on $y_i(\mu_{ln})$ hold for any $l \geq 1$:*

$$\left(\frac{1}{n}\right)^{\lambda_l - 1} \leq y_i(\mu_{ln}) \leq n. \tag{2.10}$$

Proof. We observe that for $l \geq 1$,

$$\mathbf{y}(\mu_{ln}) = \mathbf{W}^{\mu_{ln}-1:0}\mathbf{1}. \tag{2.11}$$

By Lemma 1, the matrix $\mathbf{W}^{\mu_{ln}-1:\mu_{(l-1)n}}$ is strictly positive with it's elements at least $(1/n)^{\lambda_l}$. Hence $\mathbf{W}^{\mu_{ln}-1:0}$ is the product of a strictly positive column stochastic matrix and other column stochastic matrices; consequently each of its entries are at least $(1/n)^{\lambda_l}$. Using (2.11) we derive the left part of (2.10).

Since $y_j(k) > 0$ for all j and k, using (2.9), the right part of (2.10) is concluded. □

Now we can proceed with the proof of Proposition 2.

Proof of Proposition 2. We start by rewriting the evolution of $\mathbf{z}(k)$ in a matrix form. The method to accomplish this is based on an observation from Seneta [2006]. Using (2.7), we have $x_i(k) = z_i(k)y_i(k)$ and therefore,

$$z_i(k+1)y_i(k+1) = \sum_{j=1}^{n} W_{ij}^k z_j(k)y_j(k),$$

or

$$z_i(k+1) = \sum_{j=1}^{n} \left(y_i(k+1)\right)^{-1} W_{ij}^k z_j(k)y_j(k), \tag{2.12}$$

where in the last step we used the fact that $y_i(k) \neq 0$, which is true by Lemma 5. Define,

$$\mathbf{P}^k = \left(\mathbf{Y}^{k+1}\right)^{-1} \mathbf{W}^k \mathbf{Y}^k, \tag{2.13}$$

where $\mathbf{Y}^k = \operatorname{diag}(\mathbf{y}(k))$. Using (2.12) we have,

$$\mathbf{z}(k+1) = \mathbf{P}^k \mathbf{z}(k).$$

Moreover, $\mathbf{P}^k$ is stochastic:

$$\begin{aligned}\mathbf{P}^k \mathbf{1} &= \left(\mathbf{Y}^{k+1}\right)^{-1} \mathbf{W}^k \mathbf{Y}^k \mathbf{1} = \left(\mathbf{Y}^{k+1}\right)^{-1} \mathbf{W}^k \mathbf{y}(k) \\ &= \left(\mathbf{Y}^{k+1}\right)^{-1} \mathbf{y}(k+1) = \mathbf{1}.\end{aligned}$$

Using (2.13), we obtain

$$\mathbf{P}^{\mu_{kn}-1:\mu_{(k-1)n}} = \left(\mathbf{Y}^{\mu_{kn}}\right)^{-1} \mathbf{W}^{\mu_{kn}-1:\mu_{(k-1)n}} \mathbf{Y}^{\mu_{(k-1)n}}. \tag{2.14}$$

By Lemma 1 the matrix $\mathbf{W}^{\mu_{kn}-1:\mu_{(k-1)n}}$ is strictly positive; therefore using (2.10) and (2.14), $\mathbf{P}^{\mu_{kn}-1:\mu_{(k-1)n}}$ is a strictly positive matrix with its elements at least

$$\alpha_k = \frac{1}{n}\left(\frac{1}{n}\right)^{\lambda_k}\left(\frac{1}{n}\right)^{\lambda_{k-1}-1} = \left(\frac{1}{n}\right)^{\lambda_k+\lambda_{k-1}}.$$

Using Lemma 2 we obtain,

$$z_{\max}(\mu_{kn}) - z_{\min}(\mu_{kn}) \leq (1 - n\alpha_k)\left(z_{\max}(\mu_{(k-1)n}) - z_{\min}(\mu_{(k-1)n})\right),$$

and consequently,

$$z_{\max}(\mu_{ln}) - z_{\min}(\mu_{ln}) \leq \prod_{k=1}^{l}(1 - n\alpha_k)\left(z_{\max}(0) - z_{\min}(0)\right). \tag{2.15}$$

Moreover,

$$\begin{aligned}\sum_{k=1}^{\infty} n\alpha_k \geq \sum_{k=K}^{\infty} n\alpha_k &= \sum_{k=K}^{\infty} n\left(\frac{1}{n}\right)^{\lambda_k+\lambda_{k-1}} \\ &\geq \sum_{k=K}^{\infty} n\left(\frac{1}{n}\right)^{\frac{\ln(k+T)}{2\ln(n)}+\frac{\ln(k-1+T)}{2\ln(n)}} \\ &\geq \sum_{k=K}^{\infty} n\left(\frac{1}{n}\right)^{\frac{\ln(k+T)}{\ln(n)}} \\ &= \sum_{k=K}^{\infty} \frac{n}{k+T} = \infty.\end{aligned}$$

Hence using Lemma 4 and (2.15), $z_{\max}(\mu_{ln}) - z_{\min}(\mu_{ln})$ converges to zero as $l \to \infty$. By Lemma 3 we conclude that $\lim_{k\to\infty} z_i(k)$ exists and we denote it by z_∞. We have,

$$\begin{aligned}z_\infty &= z_\infty \lim_{k\to\infty}\left(\frac{\sum_{i=1}^n y_i(k)}{\sum_{i=1}^n y_i(k)}\right) \\ &= \lim_{k\to\infty}\left(\frac{\sum_{i=1}^n z_i(k)y_i(k)}{n} + \frac{\sum_{i=1}^n (z_\infty - z_i(k))y_i(k)}{n}\right) \\ &= \frac{\sum_{i=1}^n x_i(k)}{n} + \lim_{k\to\infty}\left(\frac{\sum_{i=1}^n (z_\infty - z_i(k))y_i(k)}{n}\right) \\ &= \frac{\sum_{i=1}^n x_i(0)}{n},\end{aligned}$$

where the last equality holds due to the sum preservation property, (2.8). □

2.5 Robust asynchronous push-sum

In this section we introduce the Robust Asynchronous Push-Sum (RAPS) algorithm for distributed average computation and provide convergence guarantees for scenarios where the communication system is asynchronous and unreliable. In an unreliable setting, communication links might fail to transmit data packets and information might get lost. Convergence results proved for this algorithm will be used later when we turn to distributed optimization.

The algorithm relies heavily on the central idea of Hadjicostis et al. [2016] which is to repeatedly broadcast sums of previous messages, to deal with message losses, delays, and asynchrony. While the algorithm in Hadjicostis et al. [2016] handles message losses in a synchronous setting, RAPS can handle delays as well as asynchronicity. Bof et al. [2017b]

has proved exponential convergence of this algorithm for the case when at each iteration only one node wakes up and transmits. Here we modify the algorithm presented by Bof et al. [2017b] and show that average consensus still holds while allowing for any subset of nodes to perform updates at each iteration, in addition to allowing for bounded delays.

Pseudo-code for the algorithm is given in the box for Algorithm 1. The pseudo-code for the algorithm may appear complicated at first glance; this is because of the considerable complexity required to deal with directed communications, message losses, delays, and asynchrony. We begin by outlining the operation of the algorithm.

Without loss of generality, we define an iteration by discretizing time into time slots indexed by $k = 0, 1, 2, \ldots$. We assume that during each time slot every agent makes at most one update and processes messages sent in previous time slots. In this algorithm, as opposed to the previous ones, we assume nodes do not have self-loops.

Algorithm 1 Robust Asynchronous Push-Sum (RAPS)

Initialize the algorithm with $\mathbf{y}(0) = \mathbf{1}$, $\phi_i^x(0) = \phi_i^y(0) = 0$, $\forall i \in \{1, \ldots, n\}$ and $\rho_{ij}^x(0) = \rho_{ij}^y(0) = 0$, $\kappa_{ij} = 0$, $\forall (j, i) \in \mathcal{E}$.
At every iteration $k = 0, 1, 2, \ldots$, for every node i:
if node i wakes up **then**
  $\kappa_i \leftarrow k$;
  $\phi_i^x \leftarrow \phi_i^x + \frac{x_i}{d_i^+ + 1}$, $\phi_i^y \leftarrow \phi_i^y + \frac{y_i}{d_i^+ + 1}$;
  $x_i \leftarrow \frac{x_i}{d_i^+ + 1}$, $y_i \leftarrow \frac{y_i}{d_i^+ + 1}$;
  Node i broadcasts $(\phi_i^x, \phi_i^y, \kappa_i)$ to its out-neighbors in N_i^+.
  Processing the received messages
  for $(\phi_j^x, \phi_j^y, \kappa_j')$ in the inbox **do**
    if $\kappa_j' > \kappa_{ij}$ **then**
      $\rho_{ij}^{*x} \leftarrow \phi_j^x$, $\rho_{ij}^{*y} \leftarrow \phi_j^y$;
      $\kappa_{ij} \leftarrow \kappa_j'$;
    end if
  end for
  $x_i \leftarrow x_i + \sum_{j \in N_i^-} \left(\rho_{ij}^{*x} - \rho_{ij}^x \right)$, $y_i \leftarrow y_i + \sum_{j \in N_i^-} \left(\rho_{ij}^{*y} - \rho_{ij}^y \right)$;
  $\rho_{ij}^x \leftarrow \rho_{ij}^{*x}$, $\rho_{ij}^y \leftarrow \rho_{ij}^{*y}$,
  $z_i \leftarrow \frac{x_i}{y_i}$;
end if
Other variables remain unchanged.

Let us provide a simple intuition behind the RAPS algorithm. Each agent i holds a value

(mass) x_i and y_i. At the beginning of every iteration, i wants to split its mass between itself and its out-neighbors $j \in N_i^+$. However, to handle message losses, it sends the accumulated x and y mass (running sums which we denote by ϕ_i^x and ϕ_i^y), that i wants to transfer to each of its neighbors, from the start of the algorithm. Therefore, when a neighbor j receives a new accumulated mass from i, it stores it at ρ_{ji}^* and by subtracting the previous accumulated mass ρ_{ji} it had received from i, j obtains all the mass that i has been trying to send since its last successful communication. Then, j updates its x and y mass by adding the new received masses, and finally, updates its estimate of the average to $z_j = x_j/y_j$.

2.5.1 Delay-free unreliable communication setting

Next, we state sufficient connectivity conditions for RASP to converge to average consensus while allowing for unreliable communication system (without delays) and slowly growing to unbounded inter-communication intervals.

Theorem 1. *Suppose we apply the Robust Asynchronous Push-Sum algorithm to a set of agents communicating with each other through a strongly connected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where $\mathcal{E}$ does not have self-loops. Let $\mathcal{G}^0, \mathcal{G}^1, \ldots,$ be the sequence of graphs $\mathcal{G}^i = (\mathcal{N}, \mathcal{E}^i)$, $\mathcal{E}^i \subset \mathcal{E}$, containing only the links which transmit successfully at iteration i. Also, suppose there is another sequence $b_1, b_2, \ldots,$ of positive integers such that, if we split the sequence of $\mathcal{G}^0, \mathcal{G}^1, \ldots,$ to consecutive blocks of length b_i, the union of graphs of each block is equal to $\mathcal{G}$; i.e., $\cup_{i=\mu_k}^{\mu_{k+1}-1} \mathcal{E}^i = \mathcal{E}, \forall k \geq 0$, where μ_k and λ_k are defined in (2.2) and (2.3). Suppose that there exists some $K \geq 1, T \geq 0$, such that $\lambda_k \leq \frac{\ln(k+T)}{6\ln(n)}$, $\forall k \geq K$. Then, $z_i(k) = x_i(k)/y_i(k)$ converges to the average of $\boldsymbol{x}(0)$, i.e.,*

$$\lim_{k\to\infty} z_i(k) = \frac{\sum_{j=1}^n x_j(0)}{n}.$$

In what follows in this section, we prove Theorem 1. Similar to the proofs of the previous propositions, here we first rewrite the evolution of $\mathbf{x}(k)$ and $\mathbf{y}(k)$ in a matrix form. We show these matrices are column stochastic. Then we write the evolution of the agents' estimate of the average, $\mathbf{z}(k)$, in matrix form. Finally, we exploit the properties of these matrices to show the convergence of $z_i(k)$ to one limit which turns out to be the average.

Before we rewrite the iteration in a matrix form, we introduce the indicator variables

$\tau_i(k)$, for $i = 1, 2, \ldots, n$, and $\tau_{ij}(k)$, for $(i,j) \in \mathcal{E}$. $\tau_i(k)$ is equal to 1 if node i wakes up at time k, and is 0 otherwise. Likewise $\tau_{ij}(k)$ is 1 whenever node i wakes up at time k, $j \in N_i^+$ and the edge (i,j) is reliable, while it is 0 otherwise. Let us introduce the following variables:

$$u_{ij}(k) = \phi_i^x(k) - \rho_{ji}^x(k), \qquad \forall (i,j) \in \mathcal{E},$$
$$v_{ij}(k) = \phi_i^y(k) - \rho_{ji}^y(k), \qquad \forall (i,j) \in \mathcal{E},$$

which are, intuitively, the total x-mass and y-mass, respectively, that has been sent by node i but due to link failures has not been delivered to node j yet. The evolution of y-mass is exactly the same as x-mass; hence to avoid repetition, we only analyze the evolution of $\mathbf{x}(k)$ and $u_{ij}(k)$. We can write the update equations:

$$u_{ij}(k+1) = \left(1 - \tau_i^k \tau_{ij}(k)\right)\left(u_{ij}(k) + \tau_i(k)\frac{x_i(k)}{d_i^+ + 1}\right), \tag{2.16}$$

$$x_i(k+1) = \sum_{j \in N_i^-}\left(\frac{x_j(k)}{d_j^+ + 1} + u_{ji}(k)\right)\tau_j(k)\tau_{ji}(k) + x_i(k)\left(1 - \tau_i(k) + \frac{\tau_i(k)}{d_i^+ + 1}\right). \tag{2.17}$$

Let us introduce the column vectors $\mathbf{u}(k)$ and $\mathbf{v}(k)$ which collect all different $u_{ij}(k)$ and $v_{ij}(k)$, respectively. Moreover, let us introduce the column vectors $\boldsymbol{\chi}(k) = \left[\mathbf{x}(k)^\top, \mathbf{u}(k)^\top\right]^\top$, $\boldsymbol{\psi}(k) = \left[\mathbf{y}(k)^\top, \mathbf{v}(k)^\top\right]^\top \in \mathbb{R}^{n+m}$, where $m = |\mathcal{E}|$. Using (2.16) and (2.17) we can rewrite the algorithm in the following matrix form:

$$\boldsymbol{\chi}(k+1) = \mathbf{M}^k \boldsymbol{\chi}(k), \tag{2.18}$$

$$\boldsymbol{\psi}(k+1) = \mathbf{M}^k \boldsymbol{\psi}(k). \tag{2.19}$$

Lemma 6. *$\boldsymbol{M}$ is column stochastic and each positive element of it is at least $1/(\max_i\{d_i^+\}+1)$. Also we have for $1 \le i \le n$:*

$$M_{ii}^k = \begin{cases} 1, & \text{if } \tau_i(k) = 0, \\ \frac{1}{d_i^+ + 1}, & \text{if } \tau_i(k) = 1. \end{cases} \tag{2.20}$$

Proof. Let us first consider the i^{th} column of $\mathbf{M}^k$, with $1 \le i \le n$. The element M_{ii}^k indicates

how $x_i(k)$ influences $x_i(k+1)$. Using (2.17), it follows:

$$M_{ii}^k = 1 - \tau_i(k) + \frac{\tau_i(k)}{d_i^+ + 1} = \begin{cases} 1, & \text{if } \tau_i(k) = 0, \\ \frac{1}{d_i^+ + 1}, & \text{if } \tau_i(k) = 1. \end{cases} \tag{2.21}$$

The element M_{ji}^k, $j \in \{1, \ldots, n\} \setminus \{i\}$ indicates how $x_i(k)$ influences $x_j(k+1)$. It holds,

$$M_{ji}^k = \begin{cases} \frac{\tau_i(k)\tau_{ij}(k)}{d_i^+ + 1}, & \text{if } j \in N_i^+, \\ 0, & \text{otherwise.} \end{cases} \tag{2.22}$$

Finally, if $h \in \{n+1, \ldots, n+m\}$ is such that $\boldsymbol{\chi}_h(k) = u_{rj}(k)$ for some r, j ; the element M_{hi}^k indicates how $x_i(k)$ influences $u_{rj}(k+1)$, we have

$$M_{hi}^k = \begin{cases} \frac{(1-\tau_{ij}(k))\tau_i(k)}{d_i^+ + 1}, & \text{if } r = i, \\ 0, & \text{otherwise.} \end{cases} \tag{2.23}$$

Using (2.21)-(2.23), entries of i^{th} column of $\mathbf{M}^k$ sum to 1.

Now we consider the h^{th} column of $\mathbf{M}^k$, $h \in \{n+1, \ldots, n+m\}$. Suppose $\boldsymbol{\chi}_h(k) = u_{ij}(k)$, we have

$$M_{jh}^k = \tau_i(k)\tau_{ij}(k), \tag{2.24}$$

$$M_{hh}^k = 1 - \tau_i(k)\tau_{ij}(k), \tag{2.25}$$

and all the other elements of h^{th} column are zero. Using (2.24) and (2.25), the entries of the h^{th} column sum to 1 and hence the matrix $\mathbf{M}^k$ is column stochastic. □

Let us augment the graph $\mathcal{G}^k$ to $\mathcal{H}^k = \mathcal{G}_{\mathbf{M}^k}$ by adding auxiliary nodes b_{ij}, $\forall (i,j) \in \mathcal{E}$. Note that by Lemma 6, node $i \in \{1, \ldots, n\}$ has self-loop all the time and node b_{ij} has self-loop unless the link (i,j) transmits reliably. Let us call nodes b_{ij} *buffers* and assign values $u_{ij}(k)$ and $v_{ij}(k)$ to them.

The algorithm is equivalent to the following process: Suppose node i wakes up. If the link (i,j) works properly, node i sends some mass ($x_i(k)/(d_i^+ + 1)$ and $y_i(k)/(d_i^+ + 1)$) to node j and also node b_{ij} sends all of its mass ($u_{ij}(k)$ and $v_{ij}(k)$) to node j and becomes zero. Otherwise, the mass is sent from node i to node b_{ij} instead of j. Then all the mass gets accumulated at node b_{ij} because of its self-loop, until the link (i,j) transmits reliably.

Lemma 7. *The first n rows of* $\boldsymbol{M}^{\mu_{l+n}-1:\mu_l}$ *are strictly positive,* $l \geq 0$*. The positive elements of this matrix are at least* $(1/n)^{\mu_{l+n}-\mu_l}$*.*

Proof. Observing $\mathcal{H}^k$, every node $j \in \{1, \ldots, n\}$ has self-loop in every iteration and buffer b_{ij} has self-loop unless link (i, j) transmits successfully. We also know that during period μ_k to $\mu_{k+1} - 1$, $k = 0, 1, \ldots$, each edge $(i, j) \in \mathcal{E}$ transmits successfully at least once. Moreover, $\mathcal{G}$ is strongly connected; Hence at the end of period μ_l to $\mu_{l+n} - 1$, every node $j \in \{1, \ldots, n\}$ is reachable from all the nodes in graph $\mathcal{H}$. Also, since each positive element of $\mathbf{M}^k$ is at least $1/n$, each positive element of $\mathbf{M}^{\mu_{l+n}-1:\mu_l}$ is at least $(1/n)^{\mu_{l+n}-\mu_l}$. □

Define $\mathbf{W}^k = \mathbf{M}^{\mu_{(k+1)n}-1:\mu_{kn}}, k \geq 0$, which has positive elements of at least $\alpha^{\lambda_{k+1}}$ where $\alpha = 1/n$. Then we have:

$$\begin{bmatrix} \mathbf{x}(\mu_{(k+1)n}) \\ \mathbf{u}(\mu_{(k+1)n}) \end{bmatrix} = \mathbf{W}^k \begin{bmatrix} \mathbf{x}(\mu_{kn}) \\ \mathbf{u}(\mu_{kn}) \end{bmatrix}, \tag{2.26}$$

$$\begin{bmatrix} \mathbf{y}(\mu_{(k+1)n}) \\ \mathbf{v}(\mu_{(k+1)n}) \end{bmatrix} = \mathbf{W}^k \begin{bmatrix} \mathbf{y}(\mu_{kn}) \\ \mathbf{v}(\mu_{kn}) \end{bmatrix}. \tag{2.27}$$

Let us split the matrix $\mathbf{W}^k$ to four sub-matrices as follows:

$$\mathbf{W}^k = \begin{bmatrix} \mathbf{A}^k & \mathbf{B}^k \\ \mathbf{C}^k & \mathbf{D}^k \end{bmatrix}, \tag{2.28}$$

where $\mathbf{A}^k \in \mathbb{R}^{n \times n}$, $\mathbf{B}^k \in \mathbb{R}^{n \times m}$, $\mathbf{C}^k \in \mathbb{R}^{m \times n}$ and $\mathbf{D}^k \in \mathbb{R}^{m \times m}$. By Lemma 7 we know that matrices $\mathbf{A}^k$ and $\mathbf{B}^k$ are strictly positive.

For $h = 1, \ldots, m$ define r_h^k as follows:

$$r_h(k) = \begin{cases} u_h(k)/v_h(k), & \text{if } v_h(k) \neq 0, \\ 0, & \text{if } v_h(k) = 0. \end{cases}$$

Lemma 8. $u_{ij}(k) = 0$ *whenever* $v_{ij}(k) = 0$*.*

Proof. Since $\mathbf{v}(0) = \mathbf{0}_m$ and $\mathbf{y}(0) = \mathbf{1}_n$ and node i has self loop in graph $\mathcal{H}^k$ for all $k \geq 0$, $y_i(k)$ is always positive. If $v_{ij}(k) = 0$, the last time the node i has woken up, the link (i, j) has worked successfully, or i has not woken up yet. In either case, node b_{ij} has no remaining (x and y) mass and $u_{ij}(k) = 0$ holds. □

Therefore, the following always holds for $h = 1, \ldots, m$:

$$u_h(k) = r_h(k) v_h(k), \tag{2.29}$$

Define $\bar{\mathbf{x}}^k = \mathbf{x}(\mu_{kn})$, $\bar{\mathbf{y}}^k = \mathbf{y}(\mu_{kn})$, $\bar{\mathbf{u}}^k = \mathbf{u}(\mu_{kn})$, $\bar{\mathbf{v}}^k = \mathbf{v}(\mu_{kn})$, $\bar{\mathbf{z}}^k = \mathbf{z}(\mu_{kn})$ and $\bar{\mathbf{r}}^k = \mathbf{r}(\mu_{kn})$. Using (2.26) and (2.28) we obtain:

$$\begin{aligned} \bar{z}_i^{k+1} \bar{y}_i^{k+1} = \bar{x}_i^{k+1} &= \sum_{j=1}^{n} A_{ij}^k \bar{x}_j^k + \sum_{j=1}^{m} B_{ij}^k \bar{u}_j^k \\ &= \sum_{j=1}^{n} A_{ij}^k \bar{z}_j^k \bar{y}_j^k + \sum_{j=1}^{m} B_{ij}^k \bar{r}_j^k \bar{v}_j^k. \end{aligned}$$

Hence,

$$\begin{aligned} \bar{z}_i^{k+1} &= \left(\bar{y}_i^{k+1}\right)^{-1} \sum_{j=1}^{n} A_{ij}^k \bar{z}_j^k \bar{y}_j^k + \left(\bar{y}_i^{k+1}\right)^{-1} \sum_{j=1}^{m} B_{ij} \bar{r}_j^k \bar{v}_j^k, \\ \bar{\mathbf{z}}^{k+1} &= \left(\mathbf{Y}^{k+1}\right)^{-1} \mathbf{A}^k \mathbf{Y}^k \bar{\mathbf{z}}^k + \left(\mathbf{Y}^{k+1}\right)^{-1} \mathbf{B}^k \mathbf{V}^k \bar{\mathbf{r}}^k, \end{aligned}$$

where $\mathbf{Y}^k = \operatorname{diag}\left(\bar{\mathbf{y}}^k\right)$ and $\mathbf{V}^k = \operatorname{diag}\left(\bar{\mathbf{v}}^k\right)$. Note that $\bar{\mathbf{y}}^k$ is strictly positive. Similarly, using (2.27)-(2.29) we have,

$$\begin{aligned} \bar{r}_i^{k+1} \bar{v}_i^{k+1} = \bar{u}_i^{k+1} &= \sum_{j=1}^{n} C_{ij}^k \bar{x}_j^k + \sum_{j=1}^{m} D_{ij}^k \bar{u}_j^k \\ &= \sum_{j=1}^{n} C_{ij}^k \bar{z}_j^k \bar{y}_j^k + \sum_{j=1}^{m} D_{ij}^k \bar{r}_j^k \bar{v}_j^k. \end{aligned}$$

Here $\bar{\mathbf{v}}^k$, as opposed to $\bar{\mathbf{y}}^k$, is not necessarily strictly positive. Therefore instead of $\left(\mathbf{V}^k\right)^{-1}$, we define the following:

$$\tilde{v}_i^k = \begin{cases} 1/\bar{v}_i^k, & \text{if } \bar{v}_i^k \neq 0, \\ 0, & \text{if } \bar{v}_i^k = 0. \end{cases}$$

It follows:

$$\bar{r}_i^{k+1} = \tilde{v}_i^{k+1} \sum_{j=1}^{n} C_{ij}^k \bar{z}_j^k \bar{y}_j^k + \tilde{v}_i^{k+1} \sum_{j=1}^{m} D_{ij}^k \bar{r}_j^k \bar{v}_j^k,$$
$$\bar{\mathbf{r}}^{k+1} = \tilde{\mathbf{V}}^{k+1} \mathbf{C}^k \mathbf{Y}^k \bar{\mathbf{z}}^k + \tilde{\mathbf{V}}^{k+1} \mathbf{D}^k \mathbf{V}^k \bar{\mathbf{r}}^k.$$

where $\tilde{\mathbf{V}}^k = \text{diag}(\tilde{v}^k)$. Thus,

$$\begin{bmatrix} \bar{\mathbf{z}}^{k+1} \\ \bar{\mathbf{r}}^{k+1} \end{bmatrix} = \mathbf{P}^k \begin{bmatrix} \bar{\mathbf{z}}^k \\ \bar{\mathbf{r}}^k \end{bmatrix}, \tag{2.30}$$

where,

$$\mathbf{P}^k = \begin{bmatrix} \left(\mathbf{Y}^{k+1}\right)^{-1} \mathbf{A}^k \mathbf{Y}^k & \left(\mathbf{Y}^{k+1}\right)^{-1} \mathbf{B}^k \mathbf{V}^k \\ \tilde{\mathbf{V}}^{k+1} \mathbf{C}^k \mathbf{Y}^k & \tilde{\mathbf{V}}^{k+1} \mathbf{D}^k \mathbf{V}^k \end{bmatrix}. \tag{2.31}$$

Now we show that the sum of the elements of each row 1 to n of $\mathbf{P}^k$ is equal to 1, but for the rest of the rows they either sum to 1 or they are all zeros.

$$\mathbf{P}^k \begin{bmatrix} \mathbf{1}_n \\ \mathbf{1}_m \end{bmatrix} = \begin{bmatrix} \left(\mathbf{Y}^{k+1}\right)^{-1} \left(\mathbf{A}^k \bar{\mathbf{y}}^k + \mathbf{B}^k \bar{\mathbf{v}}^k\right) \\ \tilde{\mathbf{V}}^{k+1} \left(\mathbf{C}^k \bar{\mathbf{y}}^k + \mathbf{D}^k \bar{\mathbf{v}}^k\right) \end{bmatrix} = \begin{bmatrix} \left(\mathbf{Y}^{k+1}\right)^{-1} \bar{\mathbf{y}}^{k+1} \\ \tilde{\mathbf{V}}^{k+1} \bar{\mathbf{v}}^{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{1}_n \\ 1 \text{ or } 0 \\ \vdots \\ 1 \text{ or } 0 \end{bmatrix}.$$

The $(n+h)^{th}$ row of $\mathbf{P}^k$ is zero if and only if $\bar{v}_h^{k+1}$ is zero.

Lemma 9. *For $k \geq 0$ and $1 \leq i \leq n$ we have:*

$$\alpha^{\lambda_k} \leq \bar{y}_i^k \leq n. \tag{2.32}$$

Moreover, for $1 \leq h \leq m$ and $k \geq 1$ we have either $\bar{v}_h^k = 0$ or,

$$\alpha^{\lambda_k + \lambda_{k-1}} \leq \bar{v}_h^k \leq n. \tag{2.33}$$

Proof. We have for $k \geq 1$,

$$\begin{bmatrix} \bar{\mathbf{y}}^k \\ \bar{\mathbf{v}}^k \end{bmatrix} = \mathbf{W}^{k-1:0} \begin{bmatrix} \mathbf{1}_n \\ \mathbf{0}_m \end{bmatrix},$$

where $\mathbf{W}^{k-1:0}$ is the product of $\mathbf{W}^{k-1}$ and other column stochastic matrices. By Lemma 7, $\mathbf{W}^{k-1}$ has positive first n rows and its positive entries are at least α^{λ_k}. Hence $\mathbf{W}^{k-1:0}$ has positive first n rows and its positive elements are at least α^{λ_k}. We obtain for $1 \leq i \leq n$,

$$\bar{y}_i^k \geq \alpha^{\lambda_k}, \text{ for } k \geq 1.$$

Also since $\lambda_0 = 0$, $\bar{y}_i^0 = 1 = \alpha^{\lambda_0}$.

Suppose node h is the buffer of link (i, j). If $\bar{v}_h^k$ is positive for some $k \geq 0$, it is because the last time node i has woken up, link (i, j) has failed and node i has sent some value to h. Hence $W_{hi}^{k-1} \geq \alpha^{\lambda_k}$, and it follows,

$$\bar{v}_h^k \geq \alpha^{\lambda_k} \bar{y}_i^{k-1} \geq \alpha^{\lambda_k + \lambda_{k-1}}.$$

Also, due to some preservation property, we have $\bar{y}_i^k, \bar{v}_h^k \leq n$, for all i, h and k. □

Now we are able to find a lower bound on positive elements of $\mathbf{P}^k$. Let us divide $\mathbf{P}^k$ to four sub-matrices as:

$$\mathbf{P}^k = \begin{bmatrix} \mathbf{E}^k & \mathbf{F}^k \\ \mathbf{G}^k & \mathbf{H}^k \end{bmatrix},$$

where $\mathbf{E}^k \in \mathbb{R}^{n \times n}$, $\mathbf{F}^k \in \mathbb{R}^{n \times m}$, $\mathbf{G}^k \in \mathbb{R}^{m \times n}$ and $\mathbf{H}^k \in \mathbb{R}^{m \times m}$ are defined as in (2.31).

By construction, positive elements of $\mathbf{E}^k$ and $\mathbf{G}^k$ are at least $\frac{1}{n}\alpha^{\lambda_{k+1}}\alpha^{\lambda_k} = \alpha^{\lambda_{k+1}+\lambda_k+1}$. Similarly, positive elements of $\mathbf{F}^k$ and $\mathbf{H}^k$ are at least $\alpha^{\lambda_{k+1}+\lambda_k+\lambda_{k-1}+1}$. Hence we can define the following lower bound for all positive elements of $\mathbf{P}^k$:

$$\beta_k = \alpha^{\lambda_{k+1}+\lambda_k+\lambda_{k-1}+1}. \tag{2.34}$$

We note the following facts by observing (2.31):

• $\mathbf{E}^k$ is strictly positive.

• if $\bar{v}_h^k$ is positive, the h^{th} column of $\mathbf{F}^k$ is strictly positive. Otherwise the whole $(n+h)^{th}$ column of $\mathbf{P}^k$ is zero.

• if $\bar{v}_h^{k+1}$ is positive, the h^{th} row of $\mathbf{G}^k$ has at least one positive entry. This is true because during the time μ_{kn} to $\mu_{(k+1)n} - 1$, the corresponding link (i, j), transmits successfully at least once, which sets the values of $\bar{v}_h$ and $\bar{u}_h$ to 0. Therefore since $\bar{v}_h^{k+1}$ is positive, link (i, j) has failed at least once after the last successful transmission. Hence, C_{hi}^k is positive, and therefore G_{hi}^k is also positive.

Now we are ready to present the final part of the proof.

Proof of Theorem 1. Define the index set $I^k = \{h | \bar{v}_h^k > 0\}$. If $h \notin I^k$ we have $\bar{r}_h^k = \bar{v}_h^k = 0$,

and also the $(n+h)^{th}$ column of $\mathbf{P}^k$ has only zero entries; hence, $\bar{r}_h^k$ does not influence any variable of time $k+1$. We also have for $h \notin I^{k+1}$ the $(n+h)^{th}$ row of $\mathbf{P}^k$ has only zero entries. Thus, $\bar{r}_h^{k+1}$ is formed by the sum of zero numbers. Intuitively, this means that for $h \notin I^k$, $\bar{r}_h^k$ is zero and so are the coefficients related to it in (2.30). Therefore it gives us no meaningful information and it can be ignored. For the rest of the proof, we assume that all the variables $\bar{r}_h^k$ considered in the equations are the ones with $h \in I^k$.

We obtain:

$$\begin{aligned}\bar{r}_{\max}^{k+1} &\leq \beta^k \bar{z}_{\max}^k + (1-\beta^k)\max\{\bar{z}_{\max}^k, \bar{r}_{\max}^k\},\\ \bar{z}_{\max}^{k+1} &\leq \beta^k \min\{\bar{z}_{\min}^k, \bar{r}_{\min}^k\} + (1-\beta^k)\max\{\bar{z}_{\max}^k, \bar{r}_{\max}^k\}.\end{aligned}$$

Then,

$$\max\{\bar{z}_{\max}^{k+1}, \bar{r}_{\max}^{k+1}\} \leq \beta^k \bar{z}_{\max}^k + (1-\beta^k)\max\{\bar{z}_{\max}^k, \bar{r}_{\max}^k\}.$$

Similarly,

$$\min\{\bar{z}_{\min}^{k+1}, \bar{r}_{\min}^{k+1}\} \geq \beta^k \bar{z}_{\min}^k + (1-\beta^k)\min\{\bar{z}_{\min}^k, \bar{r}_{\min}^k\}.$$

We also have:

$$\begin{aligned}\bar{z}_{\max}^{k+1} &\leq \beta^k \sum_{i=1}^{n} \bar{z}_i^k + (1-n\beta^k)\max\{\bar{z}_{\max}^k, \bar{r}_{\max}^k\},\\ \bar{z}_{\min}^{k+1} &\geq \beta^k \sum_{i=1}^{n} \bar{z}_i^k + (1-n\beta^k)\min\{\bar{z}_{\min}^k, \bar{r}_{\min}^k\}.\end{aligned}$$

Thus,

$$\bar{z}_{\max}^{k+1} - \bar{z}_{\min}^{k+1} \leq (1-n\beta^k)\left(\max\{\bar{z}_{\max}^k, \bar{r}_{\max}^k\} - \min\{\bar{z}_{\min}^k, \bar{r}_{\min}^k\}\right).$$

Equivalently,

$$\begin{aligned}s^{k+1} &\leq \beta^k t^k + (1-\beta^k)s^k,\\ t^{k+1} &\leq (1-n\beta^k)s^k,\end{aligned}$$

where $s^k = \max\{\bar{z}_{\max}^k, \bar{r}_{\max}^k\} - \min\{\bar{z}_{\min}^k, \bar{r}_{\min}^k\}$ and $t^k = \bar{z}_{\max}^k - \bar{z}_{\min}^k$. Observing that $0 \leq t^k \leq s^k$, we obtain:

$$\begin{aligned}s^{k+1} &\leq \beta^k(1-n\beta^{k-1})s^{k-1} + (1-\beta^k)s^k\\ &\leq \beta^k(1-n\beta^{k-1})s^{k-1} + (1-\beta^k)s^{k-1}\\ &= (1-n\beta^k\beta^{k-1})s^{k-1}.\end{aligned}$$

Hence $\lim_{k\to\infty} s^k = 0$ if $\prod_{k=1}^{\infty}\left(1-n\beta^{2k}\beta^{2k-1}\right) = 0$, which, by Lemma 4, holds true if and

only if $\sum_{k=1}^{\infty} \beta^{2k}\beta^{2k-1} = \infty$. Using (2.34), we have:

$$\begin{aligned}\sum_{k=1}^{\infty} \beta^{2k}\beta^{2k-1} &= \sum_{k=1}^{\infty} \alpha^{\lambda_{2k+1}+2\lambda_{2k}+2\lambda_{2k-1}+\lambda_{2k-2}+2} \\ &\geq \frac{1}{n^2}\sum_{k=K}^{\infty} \alpha^{-\frac{\ln(2k+1+T)}{\ln(\alpha)}} \\ &= \frac{1}{n^2}\sum_{k=K}^{\infty} \frac{1}{2k+1+T} = \infty.\end{aligned}$$

Hence $\max\{\bar{z}^k_{\max}, \bar{r}^k_{\max}\} - \min\{\bar{z}^k_{\min}, \bar{r}^k_{\min}\}$ converges to 0 as k goes to infinity. Combining this with Lemma 3 we obtain,

$$\lim_{k\to\infty} \bar{z}^k_i = \lim_{k\to\infty,\ h\in I^k} \bar{r}^k_h = L. \tag{2.35}$$

We have:

$$\begin{aligned}L &= L \lim_{k\to\infty} \frac{\sum_{i=1}^n \bar{y}^k_i + \sum_{h=1}^m \bar{v}^k_h}{\sum_{i=1}^n \bar{y}^k_i + \sum_{h=1}^m \bar{v}^k_h} \\ &= \lim_{k\to\infty} \left(\frac{\sum_{i=1}^n \bar{z}^k_i\bar{y}^k_i + \sum_{h=1}^m \bar{r}^k_h\bar{v}^k_h}{n}\right) + \lim_{k\to\infty} \left(\frac{\sum_{i=1}^n (L-\bar{z}^k_i)\bar{y}^k_i + \sum_{h=1}^m (L-\bar{r}^k_h)\bar{v}^k_h}{n}\right) \\ &= \lim_{k\to\infty} \left(\frac{\sum_{i=1}^n \bar{x}^k_i + \sum_{h=1}^m \bar{u}^k_h}{n}\right) + \lim_{k\to\infty} \left(\frac{\sum_{i=1}^n (L-\bar{z}^k_i)\bar{y}^k_i + \sum_{h=1}^m (L-\bar{r}^k_h)\bar{v}^k_h}{n}\right) \\ &= \frac{\sum_{i=1}^n x^0_i}{n},\end{aligned}$$

where in the last equality, we used (2.35), and the fact that $\bar{v}^k_h = 0$ for $h \notin I^k$. □

2.5.2 Communication setting with bounded delays and message losses

In the remainder of this section, we provide another analysis of the RAPS algorithm, showing that it converges geometrically to the average in the presence of bounded message losses and delays, asynchronous updates, and directed communication. To handle the message delays, next, we will state a new assumption on connectivity, asynchronicity, and message loss.

Assumption 2. *(a) Graph $\mathcal{G}$ is strongly connected and does not have self-loops.*

(b) The delays on each link are bounded above by some $\Gamma_{\text{del}} \geq 1$.

(c) Every agent wakes up and performs updates at least once every $\Gamma_u \geq 1$ iterations.

(d) Each link fails at most $\Gamma_f \geq 0$ consecutive times.

(e) Messages arrive in the order of time they were sent. In other words, if messages are sent from node i to j at times k_1 and k_2 with (effective) delays d_1 and d_2, respectively, and $k_1 < k_2$, then we have $k_1 + d_1 < k_2 + d_2$.

One consequence of Assumption 2 is that the effective delays associated with each message that gets through are bounded above by $\Gamma_d := \Gamma_{\text{del}} + \Gamma_u - 1$. Another consequence is that for each $(i, j) \in \mathcal{E}$, j receives a message from i successfully, at least once every Γ_s iterations where

$$\Gamma_s := \Gamma_u(\Gamma_f + 1) + \Gamma_d \geq 2. \tag{2.36}$$

Part (e) of Assumption 2 can be assumed without loss of generality. Indeed, observe that outdated messages automatically get discarded in Line 10 of our algorithm.

The main result of this subsection is exponential convergence of RAPS to initial average, stated next.

Theorem 2. *Suppose Assumption 2 holds. Then RAPS converges exponentially to the initial mean of agent values. i.e.,*

$$\left| z_i(k) - \frac{1}{n}\sum_{i=1}^{n} x_i(0) \right| \leq \delta\lambda^k \|\mathbf{x}(0)\|_1,$$

where $\delta := \frac{1}{1-n\alpha^6}$, $\lambda := (1 - n\alpha^6)^{1/(2n\Gamma_s)}$ and $\alpha := (1/n)^{n\Gamma_s}$.

It is worth mentioning that even though $1/(1-\lambda) = \mathcal{O}(n^{p(n)})$ where $p(n) = \mathcal{O}(n)$, this is a bound for a worst case scenario and on average, as it can be seen in numerical simulations, RAPS performs better. Moreover, when the graph $\mathcal{G}$ satisfies certain properties, such as regularity, and also there is no link delays and failures, we have $1/(1-\lambda) = \mathcal{O}(n^3)$ (see Nedić and Olshevsky [2016, Theorem 1]). More broadly, that paper establishes that $1/(1-\lambda)$ will scale with the mixing rate of the underlying Markov process.

Next, we will prove Theorem 2. Our first step is to formulate the RAPS algorithm in terms of a linear update (i.e., a matrix multiplication). Let us introduce the following indicator variables (similar to ones in the previous subsection): $\tau_i(k)$ for $i \in \{1, \ldots, n\}$

which equals to 1 if node i wakes up at time k, and equals 0 otherwise. Similarly, $\tau_{ij}^l(k)$ for $(i,j) \in \mathcal{E}$, $1 \leq l \leq \Gamma_d$, which is 1 if $\tau_i(k) = 1$ **and** the message sent from node i to j at time k will arrive after experiencing an effective delay of l. [1] Note that if node i wakes up at time k but the message it sends to j is lost, then $\tau_{ij}^l(k)$ will be zero for all l.

We can rewrite the RAPS algorithm with the help of these indicator variables. Let us adopt the notation that $x_i(k)$ refers to x_i at the **beginning** of round k of the algorithm (i.e., before node i has a chance to go through the list of steps outlined in the algorithm box). We will use the same convention with all of the other variables, e.g., $y_i(k), z_i(k)$, etc. If node i does not wake up at round k, then of course $x_i(k+1) = x_i(k)$.

Now observe that we can write

$$\phi_i^x(k+1) - \phi_i^x(k) = \tau_i(k) \frac{x_i(k)}{d_i^+ + 1}. \tag{2.37}$$

Likewise, we have

$$x_i(k+1) = x_i(k) \left(1 - \tau_i(k) + \frac{\tau_i(k)}{d_i^+ + 1}\right) + \sum_{j \in N_i^-} \left(\rho_{ij}^x(k+1) - \rho_{ij}^x(k)\right), \tag{2.38}$$

which can be shown by considering each case ($\tau_i(k) = 1$ or 0); note that we have used the fact that, in the event that node i wakes up at time k, the variable $\rho_{ij}^x(k+1)$ equals the variable ρ_{ij}^{*x} during execution of Line 16 of the algorithm at time k.

Finally, we have that $\forall (i,j) \in \mathcal{E}$, the flows ρ_{ji}^x are updated as follows:

$$\rho_{ji}^x(k+1) = \rho_{ji}^x(k) + \sum_{l=1}^{\Gamma_d} \tau_{ij}^l(k-l) \left(\phi_i^x(k+1-l) - \rho_{ji}^x(k)\right), \tag{2.39}$$

where we make use of the fact that the sum contains only a single nonzero term, since the messages arrive monotonically. To parse the indices in this equation, note that node i actually broadcasts $\phi_i^x(k+1-l)$ in our notation at iteration $k-l$; by our definitions, $\phi_i^x(k-l)$ is the value of ϕ_i^x at the **beginning** of that iteration. To simplify these relations, we introduce the auxiliary variables u_{ij}^x for all $(i,j) \in \mathcal{E}$, defined through the following

[1] Note the difference between indexing in τ_{ij}^l and ρ_{ji}^x, which are both defined for link $(i,j) \in \mathcal{E}$.

recurrence relation:

$$u_{ij}^x(k+1) := \left(1 - \sum_{l=1}^{\Gamma_d} \tau_{ij}^l(k)\right)\Big(u_{ij}^x(k) + \phi_i^x(k+1) - \phi_i^x(k)\Big), \tag{2.40}$$

and initialized as $u_{ij}^x(0) := 0$. Intuitively, the variables u_{ij}^x represent the "excess mass" of x_i that is yet to reach node j. Indeed, this quantity resets to zero whenever a message is sent that arrives at some point in the future, and otherwise is incremented by adding the broadcasted mass that is lost. Note that node i never knows $u_{ij}^x(k)$, since it has no idea which messages are lost, and which are not; nevertheless, for purposes of analysis, nothing prevents us from considering these variables.

Let us also define the related quantity,

$$v_{ij}^x(k) := u_{ij}^x(k) + \phi_i^x(k+1) - \phi_i^x(k), \qquad \text{for } k \geq 0,$$

and $v_{ij}^x(k) := 0$ for $k < 0$. Intuitively, this quantity may be thought of as a forward-looking estimate of the mass that *will arrive* at node j, if the message sent from node i at time k gets through; correspondingly, it includes not only the previous unsent mass, but the extra mass that will be added at the current iteration.

The key variables for the analysis of our method are the variables we will denote by $x_{ij}^l(k)$. Intuitively, every time a message is sent, but gets lost, we imagine that it has instead arrived into a "virtual node" which holds that mass; once the next message gets through, we imagine that the virtual node has forwarded that mass to its intended destination. This idea originates from Hadjicostis et al. [2016]. Because of the delays, however, we need to introduce Γ_d virtual nodes for each such event. If a message is sent from i and arrives at j with effective delay l, we will instead imagine it is received by the virtual node b_{ij}^l, then sent to b_{ij}^{l-1} at the next time step, and so forth until it reaches b_{ij}^1, and is then forwarded to its destination. These virtual nodes are defined formally later.

Putting that intuition aside, we formally define the variables $x_{ij}^l(k)$ via the following set

of recurrence relations:

$$x_{ij}^{l}(k+1) := \tau_{ij}^{l}(k)\upsilon_{ij}^{x}(k), \qquad l = \Gamma_d, \tag{2.41}$$

$$x_{ij}^{l}(k+1) := \tau_{ij}^{l}(k)\upsilon_{ij}^{x}(k) + x_{ij}^{l+1}(k), \qquad 1 \leq l < \Gamma_d, \tag{2.42}$$

and $x_{ij}^{l}(k) := 0$ when both $k \leq 0$ and $l = 1, \ldots, \Gamma_d$. To parse these equations, imagine what happens when a message is sent from i to j with effective delay of Γ_d at time k. The content of this message becomes the value of $x_{ij}^{\Gamma_d}$ according to (2.41); and, in each subsequent step, influences $x_{ij}^{\Gamma_d-1}, x_{ij}^{\Gamma_d-2}$, and so forth according to (2.42). Putting (2.41) and (2.42) together, we obtain

$$x_{ij}^{l}(k) = \sum_{t=1}^{\Gamma_d-l+1} \tau_{ij}^{t+l-1}(k-t)\upsilon_{ij}^{x}(k-t), \tag{2.43}$$

and particularly,

$$x_{ij}^{1}(k) = \sum_{t=1}^{\Gamma_d} \tau_{ij}^{t}(k-t)\upsilon_{ij}^{x}(k-t). \tag{2.44}$$

Note that, as is common in many of the equations we will write, only a single term in the sums can be nonzero (this is not obvious at this point and is a result of Lemma 10).

Before proceeding to the main result of this section, we state the following lemma, whose proof is immediate.

Lemma 10. *If $\tau_{ij}^{l}(k) = 1$, the following statements are satisfied:*

(a) $\tau_{ij}^{l'}(k) = 0$ for $l' \neq l$.

(b) If $l > 0$, then $\tau_{ij}^{s}(k+t) = 0$ for $t = 1, \ldots, l$ and $s = 0, \ldots, l-t$.

(c) If $l < \Gamma_d$, then $\tau_{ij}^{s}(k-t) = 0$ for $t = 1, \ldots, \Gamma_d - l$ and $s = l+t, \ldots, \Gamma_d$.

Lemma 11. *If $\tau_{ij}^{l}(k) = 1$ then $x_{ij}^{l'}(k) = 0$ for $l' > l$.*

Proof. By Lemma 10(c), $\tau_{ij}^{t+l'-1}(k-t) = 0$ for $t \in \{1, \ldots, \Gamma_d - l' + 1\}$. Hence, by (2.43) we have,

$$x_{ij}^{l'}(k) = \sum_{t=1}^{\Gamma_d-l'+1} \tau_{ij}^{t+l'-1}(k-t)\upsilon_{ij}^{x}(k-t) = 0.$$

□

The next lemma is essentially a restatement of the observation that the content of every $x_{ij}^{l'}$ eventually "passes through" x_{ij}^{1}.

Lemma 12. *If $\tau_{ij}^{l}(k-l)=1$, $l\geq 1$, we have,*

$$\sum_{l'=1}^{l} x_{ij}^{l'}(k-l)=\sum_{t=1}^{l} x_{ij}^{1}(k-t).$$

Proof. We will show $x_{ij}^{1}(k-t)=x_{ij}^{l-t+1}(k-l)$ for $t=1,\ldots,l$. For $t=l$ the equality is trivial. Now suppose $t<l$. By Lemma 10(a) we have $\tau_{ij}^{l-t}(k-l)=0$. Moreover, by part (b) of the same lemma we have, $\tau_{ij}^{s'}(k-l+t')=0$ for $t'=1,\ldots,l-t-1$ and $s'=l-t-t'$. Hence, $x_{ij}^{l-t-t'+1}(k-l+t')=x_{ij}^{l-t-t'}(k-l+t'+1)$. Combining these equations for $t'=0,\ldots,l-t-1$, we get $x_{ij}^{1}(k-t)=x_{ij}^{l-t+1}(k-l)$. □

The following lemma is the key step of a linear formulation of RAPS.

Lemma 13. *For $k=0,1,\ldots$ and $(i,j)\in\mathcal{E}$ we have:*

$$\rho_{ji}^{x}(k+1)-\rho_{ji}^{x}(k)=x_{ij}^{1}(k), \tag{2.45}$$

$$u_{ij}^{x}(k+1)+\rho_{ji}^{x}(k+1)+\sum_{l=1}^{\Gamma_d} x_{ij}^{l}(k+1)=\phi_{i}^{x}(k+1). \tag{2.46}$$

Parsing these equations, (2.45) simply states that the value of $x_{ij}^{1}(k)$ can be thought of as impacting ρ_{ji}^{x} at time k; recall that the content of $x_{ij}^{1}(k)$ is a message that was sent from node i to j at time $k-l$ with an effective delay of l, for some $1\leq l\leq\Gamma_d$ (cf. (2.44)). On the other hand, (2.46) may be thought of a "conservation of mass" equation. All the mass that has been sent out by node i has either: (i) been lost (in which case it is in u_{ij}^{x}), (ii) affected node j (in which case it is in ρ_{ji}^{x}), or (iii) is in the process of reaching node j but delayed (in which case it is in some x_{ij}^{l}). Although this lemma is arguably obvious, a formal proof is surprisingly lengthy. Nontheless, we have provided the proof below:

Proof of Lemma 13. We use mathematical induction. For $k=0$ we have $x_{ij}^{l}(0)=0,\ \forall l$ and

$u_{ij}^x(0) = \phi_i^x(0) = \rho_{ji}^x(0) = 0$. By (2.39) and the definition of u_{ij}^x and x_{ij}^l we obtain,

$$\begin{aligned}
\rho_{ji}^x(1) &= 0, \\
u_{ij}^x(1) &= (1 - \sum_{l=1}^{\Gamma_d} \tau_{ij}^l(0))\phi_i^x(1), \\
\sum_{l=1}^{\Gamma_d} x_{ij}^l(1) &= (\sum_{l=1}^{\Gamma_d} \tau_{ij}^l(0))\phi_i^x(1).
\end{aligned}$$

Equation (2.45) is concluded from first equation above and (2.46) results by summing up all three equations above.

Now assume this lemma is true for $k = 0, \ldots, K-1$. We want to show it will be true for $k = K$ as well. In the following, LHS and RHS denote the left-hand-side and right-hand-side of (2.45) for $k = K$. By (2.39) we have,

$$LHS = \sum_{l=1}^{\Gamma_d} \tau_{ij}^l(K-l)[\phi_i^x(K+1-l) - \rho_{ji}^x(K)].$$

Using (2.44) we obtain,

$$RHS = \sum_{l=1}^{\Gamma_d} \tau_{ij}^l(K-l)\upsilon_{ij}^x(K-l).$$

Hence, it suffices to show that:

$$\sum_{l=1}^{\Gamma_d} \tau_{ij}^l(K-l)[\phi_i^x(K+1-l) - \rho_{ji}^x(K) - \upsilon_{ij}^x(K-l)] = 0. \tag{2.47}$$

By part (e) of Assumption 2, at most one of the $\tau_{ij}^l(K-l)$, $l = 1, \ldots, \Gamma_d$ is non-zero. If all are zeros, the result follows. Now suppose $\tau_{ij}^l(K-l) = 1$ for some l. Equation (2.47) becomes,

$$\phi_i^x(K+1-l) - \rho_{ji}^x(K) - \upsilon_{ij}^x(K-l) = 0.$$

Plugging in the definition of υ_{ij}^x, after rearrangement we obtain,

$$\phi_i^x(K-l) - u_{ij}^x(K-l) = \rho_{ji}^x(K). \tag{2.48}$$

By the induction hypothesis, (2.45) holds for $k = K - t$, $t = 1, \ldots, l$. Therefore,

$$\rho^x_{ji}(K+1-t) - \rho^x_{ji}(K-t) = x^1_{ij}(K-t).$$

Hence,

$$\begin{aligned}
\rho^x_{ji}(K) &= \rho^x_{ji}(K-l) + \sum_{t=1}^{l} (\rho^x_{ji}(K+1-t) - \rho^x_{ji}(K-t)) \\
&= \rho^x_{ji}(K-l) + \sum_{t=1}^{l} x^1_{ij}(K-t) \\
&= \rho^x_{ji}(K-l) + \sum_{l'=1}^{l} x^{l'}_{ij}(K-l) \qquad \text{(Lemma 12)} \\
&= \rho^x_{ji}(K-l) + \sum_{l'=1}^{d} x^{l'}_{ij}(K-l). \qquad \text{(Lemma 11)}
\end{aligned}$$

Moreover, by the induction hypothesis, (2.46) holds for $k = K - l$, thus,

$$\phi^x_i(K-l) - u^x_{ij}(K-l) = \rho^x_{ji}(K-l) + \sum_{l'=1}^{\Gamma_d} x^{l'}_{ij}(K-l).$$

Combining the two relations above we conclude (2.48).

To show (2.46), consider the following equations which are direct results of the definitions and (2.45) that we just showed for $k = K$:

$$\begin{aligned}
u^x_{ij}(K+1) &= (1 - \sum_{l=1}^{\Gamma_d} \tau^l_{ij}(K)) \upsilon^x_{ij}(K), \\
\rho^x_{ji}(K+1) &= \rho^x_{ji}(K) + x^1_{ij}(K), \\
\sum_{l=1}^{\Gamma_d} x^l_{ij}(K+1) &= \sum_{l=2}^{\Gamma_d} x^l_{ij}(K) + \sum_{l=1}^{\Gamma_d} \tau^l_{ij}(K) \upsilon^x_{ij}(K).
\end{aligned}$$

Summing up both sides of the equations above we have,

$$\begin{aligned} LHS &= u_{ij}^x(K+1) + \rho_{ji}^x(K+1) + \sum_{l=1}^{\Gamma_d} x_{ij}^l(K+1), \\ RHS &= \sum_{l=1}^{\Gamma_d} x_{ij}^l(K) + \rho_{ji}^x(K) + \upsilon_{ij}^x(K) \\ &= \sum_{l=1}^{\Gamma_d} x_{ij}^l(K) + \rho_{ji}^x(K) + u_{ij}^x(K) - \phi_i^x(K) + \phi_i^x(K+1) = \phi_i^x(K+1). \end{aligned}$$

The last equality holds because of the induction hypothesis (2.46) for $k = K-1$, hence completing the proof. □

We next write down a matrix form of our updates. As a first step, define the $(n+m') \times 1$ column vector $\boldsymbol{\chi}(k) := [\mathbf{x}(k)^\top, \mathbf{x}^1(k)^\top, \ldots, \mathbf{x}^{\Gamma_d}(k)^\top, \mathbf{u}^x(k)^\top]^\top$, where $m' := (\Gamma_d + 1)m$, $m := |\mathcal{E}|$, $\mathbf{x}(k)$ collects all $x_i(k)$, $\mathbf{x}^l(k)$ collects all $x_{ij}^l(k)$ and, $\mathbf{u}^x(k)$ collects all $u_{ij}^x(k)$. Define $\boldsymbol{\psi}(k)$ by collecting y-values similarly.

Now, we have all the tools to show the linear evolution of $\boldsymbol{\chi}(k)$. By Equations (2.37), (2.38) and (2.45) we have,

$$x_j(k+1) = x_j(k)\left(1 - \tau_j(k) + \frac{\tau_j(k)}{d_j^+ + 1}\right) + \sum_{i \in N_j^-} x_{ij}^1(k). \tag{2.49}$$

Moreover, by the definitions of x_{ij}, υ_{ij} and (2.37) it follows,

$$\begin{aligned} x_{ij}^{\Gamma_d}(k+1) &= \tau_{ij}^{\Gamma_d}(k)\left[u_{ij}^x(k) + \frac{x_i(k)}{d_i^+ + 1}\right], \\ x_{ij}^l(k+1) &= \tau_{ij}^l(k)\left[u_{ij}^x(k) + \frac{x_i(k)}{d_i^+ + 1}\right] + x_{ij}^{l+1}(k). \end{aligned} \tag{2.50}$$

Finally, by (2.37) and (2.40) we obtain,

$$u_{ij}^x(k+1) = \big(1 - \sum_{l=1}^{\Gamma_d} \tau_{ij}^l(k)\big)\Big(u_{ij}^x(k) + \tau_i(k)\frac{x_i(k)}{d_i^+ + 1}\Big). \tag{2.51}$$

Using (2.49) to (2.51) we can write the evolution of $\boldsymbol{\chi}(k)$ and $\boldsymbol{\psi}(k)$ in the following linear

form:

$$\begin{aligned}\boldsymbol{\chi}(k+1) &= \mathbf{M}(k)\boldsymbol{\chi}(k),\\ \boldsymbol{\psi}(k+1) &= \mathbf{M}(k)\boldsymbol{\psi}(k),\end{aligned} \tag{2.52}$$

where $\mathbf{M}(k) \in \mathbb{R}^{(n+m')\times(n+m')}$ is an appropriately defined matrix.

We have thus completed half of our goal: we have shown how to write RAPS as a linear update. Next, we show that the corresponding matrices are column-stochastic.

Lemma 14. *$\mathbf{M}(k)$ is column stochastic and its positive elements are at least $1/(\max_i\{d_i^+\}+1)$. Moreover, for $i = 1,\dots,n$, $M_{ii}(k)$ are positive.*

This lemma can be proved "by inspection." Indeed, $\mathbf{M}(k)$ is column stochastic if and only if, for every $\boldsymbol{\chi}(k)$, we have $\mathbf{1}^T\boldsymbol{\chi}(k+1) = \mathbf{1}^T\boldsymbol{\chi}(k)$. Thus one just needs to demonstrate that no mass is ever "lost," i.e., that a decrease/increase in the value of one node is always accompanied by an increase/decrease of the value of another node, which can be done just by inspecting the equations. A formal proof is nonetheless given next.

Proof. To show that $\mathbf{M}(k)$ is column stochastic, we study how each element of $\boldsymbol{\chi}(k)$ influences $\boldsymbol{\chi}(k+1)$.

For $i = 1,\dots,n$, the ith column of $\mathbf{M}(k)$ represents how $x_i(k)$ influences $\boldsymbol{\chi}(k+1)$. We will use (2.49) to (2.51) to find these coefficients.

First, $x_i(k)$ influences $x_i(k+1)$ with the coefficient $1-\tau_i(k)+\tau_i(k)/(d_i^+ + 1) > 0$. For $j \in N_i^+$, $x_i(k)$ influences $x_{ij}^l(k+1)$ by $\tau_{ij}^l(k)/(d_i^+ + 1)$ and $u_{ij}^x(k+1)$ with coefficient $(\tau_i(k) - \sum_{l=1}^{\Gamma_d}\tau_i(k)\tau_{ij}^l(k))/(d_i^+ + 1)$. Summing these coefficients up results in 1.

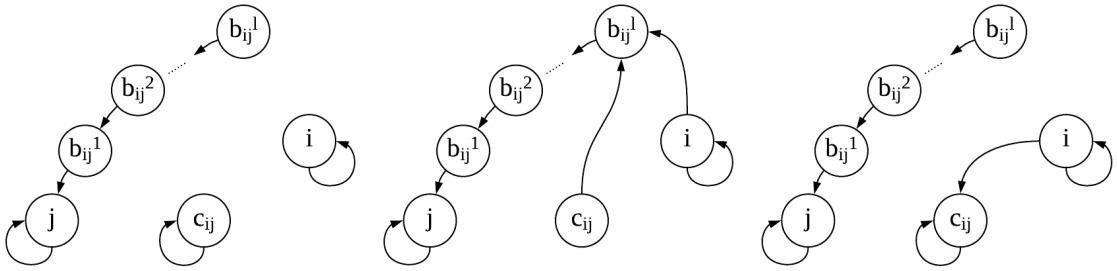
For $l = 2,\dots,\Gamma_d$, $(i,j) \in \mathcal{E}$, $x_{ij}^l(k)$ influences $x_{ij}^{l-1}(k+1)$ with coefficient 1 and $x_{ij}^1(k)$ influences $x_j(k+1)$ with coefficient 1.

Finally, $u_{ij}^x(k)$ influences $x_{ij}^l(k+1)$ with coefficient $\tau_{ij}^l(k)$ and $u_{ij}^x(k+1)$ with $(1 - \sum_{l=1}^{d}\tau_{ij}^l(k))$, which sum up to 1.

Note that all the coefficients above are at least $1/(\max_i\{d_i^+\}+1)$. □

An important result of this lemma is the sum preservation property, i.e.,

$$\begin{aligned}\sum_{i=1}^{n+m'}\chi_i(k) &= \sum_{i=1}^{n}x_i(0),\\ \sum_{i=1}^{n+m'}\psi_i(k) &= n.\end{aligned} \tag{2.53}$$



$\tau_i(k) = 0.$ $\tau_{ij}^l(k) = 1.$ $\tau_i(k) = 1,\ \tau_{ij}^l(k) = 0, \forall l.$

Figure 2·1: Augmented graph $\mathcal{H}(k)$ for different scenarios.

For further analysis, we augment the graph $\mathcal{G}$ to $\mathcal{H}(k) := \mathcal{G}_{\mathbf{M}(k)} = (\mathcal{V}_A, \mathcal{E}_A(k))$ by adding the following virtual nodes: b_{ij}^l for $l = 1, \ldots, \Gamma_d$ and $(i,j) \in \mathcal{E}$, which hold the values x_{ij}^l and y_{ij}^l; We also add the nodes c_{ij} for $(i,j) \in \mathcal{E}$ which hold the values u_{ij}^x and u_{ij}^y.

In $\mathcal{H}(k)$, there is a link from b_{ij}^l to b_{ij}^{l-1} for $1 < l \leq d$ and from b_{ij}^1 to j as they forward their values to the next node. Moreover, if $\tau_{ij}^l(k) = 1$ for some $1 \leq l \leq \Gamma_d$, then there is a link from both c_{ij} and i to b_{ij}^l.

If $\tau_{ij}^l(k) = 0$ for $1 \leq l \leq \Gamma_d$ then c_{ij} has a self loop, and if also $\tau_i(k) = 1$, there's a link from i to c_{ij}. All non-virtual agents $i \in \mathcal{V}$, have self-loops all the time (see Fig. 2·1).

Recursions (2.52) and Lemma 14 may thus be interpreted as showing that the RAPS algorithm can be thought of as a push-sum algorithm over the augmented graph sequence $\{\mathcal{H}(k)\}$, where each agent (virtual and non-virtual) holds an x-value and a y-value which evolve similarly and in parallel.

Unfortunately, Theorem 2 does not follow immediately from standard results on exponential convergence of push-sum. The reason is that the connectivity conditions assumed for such theorems are not satisfied here: there will not always be paths leading to virtual nodes from non-virtual nodes. Nevertheless, with some suitable modifications, the existence of paths from virtual nodes to other virtual nodes is sufficient, as we will show next.

Before proving the theorem, we need the following lemmas and definitions. Given a sequence of graphs $\mathcal{G}^0, \mathcal{G}^1, \mathcal{G}^2, \ldots$, we will say node b is reachable from node a in time period k_1 to k_2 $(k_1 < k_2)$, if there exists a sequence of directed edges $e_{k_1}, e_{k_1+1}, \ldots, e_{k_2}$ such that

e_k is in $\mathcal{G}^k$, the destination of e_k is the origin of e_{k+1} for $k_1 \leq k < k_2$, and the origin of e_{k_1} is a and the destination of e^{k_2} is b.

Our first lemma provides a standard lower bound on the entries of the column-stochastic matrices from (2.52).

Lemma 15. $\mathbf{M}^{k+n\Gamma_s-1:k}$ *has positive first* n *rows, for any* $k \geq 0$. *The positive elements of this matrix are at least*

$$\alpha = (1/n)^{n\Gamma_s}.$$

Proof. By Lemma 14, each node $j \in \mathcal{V}$ has self-loops at every iteration in the augmented graph $\mathcal{H}$. Since $\mathcal{G}$ is strongly connected, the set of reachable non-virtual nodes from any node $a_h \in \mathcal{V}_A$ strictly increases every Γ_s iterations. Hence, $\mathbf{M}^{k+n\Gamma_s-1:k}$ has positive first n rows. Moreover, since all positive elements of M are at least $1/n$, the positive elements of $\mathbf{M}^{k+n\Gamma_s-1:k}$ are at least $(1/n)^{n\Gamma_s}$. □

Next, we give a reformulation of the push-sum update that will be key to showing the exponential convergence of the algorithm. The proof is a minor variation of Lemma 4 in Nedić and Olshevsky [2016].

Lemma 16. *Consider the vectors* $\mathbf{u}(k) \in \mathbb{R}^d$, $\mathbf{v}(k) \in \mathbb{R}^d_+$ *and square matrix* $\mathbf{A}(k) \in \mathbb{R}^{d\times d}_+$, *for* $k \geq 0$ *such that,*

$$\begin{aligned} \mathbf{u}(k+1) &= \mathbf{A}(k)\mathbf{u}(k), \\ \mathbf{v}(k+1) &= \mathbf{A}(k)\mathbf{v}(k). \end{aligned} \tag{2.54}$$

Also suppose $u_i(k) = 0$ *if* $v_i(k) = 0$, $\forall k, i$. *Define* $\mathbf{u}^-(k) \in \mathbb{R}^d$ *as:*

$$u_i^-(k) := \begin{cases} 1/u_i(k), & \text{if } u_i(k) \neq 0, \\ 0, & \text{if } u_i(k) = 0. \end{cases}$$

Define $\mathbf{r}(k) := \mathbf{u}(k) \circ \mathbf{v}^-(k)$, *where* $\circ$ *denotes the element-wise product of two vectors. Then we have,*

$$\mathbf{r}(k+1) = \mathbf{B}(k)\mathbf{r}(k),$$

where $\mathbf{B}(k) \in \mathbb{R}^{d\times d}_+$ *is defined as,*

$$\mathbf{B}(k) := \mathrm{diag}(\mathbf{v}^-(k+1))\mathbf{A}(k)\mathrm{diag}(\mathbf{v}(k)).$$

Proof. Since $u_i(k) = 0$ if $v_i(k) = 0$, $u_i(k) = r_i(k)v_i(k)$ holds for all i, k. Substituting in (2.54) we obtain,

$$r_i(k+1)v_i(k+1) = \sum_{j=1}^{d} A_{ij}(k)r_j(k)v_j(k).$$

Since, by definition $r_i(k) = 0$ if $v_i(k) = 0$, $\forall k, i$, we get

$$r_i(k+1) = v_i^-(k+1)\sum_{j=1}^{d} A_{ij}(k)r_j(k)v_j(k).$$

Therefore,

$$\mathbf{r}(k+1) = \text{diag}(\mathbf{v}^-(k+1))\mathbf{A}(k)\text{diag}(\mathbf{v}(k))\mathbf{r}(k).$$

□

Our next corollary, which follows immediately from the previous lemma, characterizes the dichotomy inherent in push-sum with virtual nodes: every row either adds up to one or zero.

Corollary 1. *Consider the matrix* $\mathbf{B}(k)$ *defined in Lemma 16. Let us define the index set* $J^k := \{i |\, v_i(k) \neq 0\}$. *If* $i \notin J^k$, *the* i*th column of* $\mathbf{B}(k)$ *and* i*th row of* $\mathbf{B}(k-1)$ *only contain zero entries. Moreover,*

$$\begin{aligned}\mathbf{B}(k)\mathbf{1}_d &= \text{diag}(\mathbf{v}^-(k+1))\mathbf{A}(k)\mathbf{v}(k) \\ &= \text{diag}(\mathbf{v}^-(k+1))\mathbf{v}(k+1) = \begin{bmatrix} 1 \text{ or } 0 \\ \vdots \\ 1 \text{ or } 0 \end{bmatrix}.\end{aligned}$$

Hence, the i*th row of* $\mathbf{B}(k)$ *sums to* 1 *if and only if* $\mathbf{v}_i(k+1) \neq 0$ *or* $i \in J^{k+1}$.

Our next lemma characterizes the relationship between zero entries in the vectors $\boldsymbol{\chi}(k)$ and $\boldsymbol{\psi}(k)$.

Lemma 17. $\chi_h(k) = 0$ *whenever* $\psi_h(k) = 0$ *for* $h = 1, \ldots, n+m'$, $k \geq 0$.

Proof. First we note that $\boldsymbol{\psi}(0) = [\mathbf{1}_n^\top, \mathbf{0}_{m'}^\top]^\top$ and each node $i \in \mathcal{V}$ has a self-loop in graph $\mathcal{H}(k)$ for all $k \geq 0$; hence, $\psi_h(k) \geq 0$ for all h and particularly, $\psi_i(k) > 0$ for $i = 1, \ldots, n$. Now suppose $h > n$ and corresponds to a virtual agent $a_h \in \mathcal{V}_A$. If $\psi_h(k) = 0$, it means

a_h has already sent all its y-value to another node or has not received any y-value yet. In either case, that node also has no remaining x-value as well and $\chi_h(k) = 0$. □

Let us define $\boldsymbol{\psi}^-(k) \in \mathbb{R}^{n+m'}$, $k \geq 0$ by

$$\psi_i^-(k) := \begin{cases} 1/\psi_i(k), & \text{if } \psi_i(k) \neq 0, \\ 0, & \text{if } \psi_i(k) = 0. \end{cases} \tag{2.55}$$

Moreover, we define the vector $\mathbf{z}(k)$ by setting $\mathbf{z}(k) := \boldsymbol{\chi}(k) \circ \boldsymbol{\psi}^-(k)$. By (2.52) and Lemma 17, we can use Lemma 16 to obtain,

$$\mathbf{z}(k+1) = \mathbf{P}(k)\mathbf{z}(k),$$

where $\mathbf{P}(k) := \text{diag}(\boldsymbol{\psi}^-(k+1))\mathbf{M}(k)\text{diag}(\boldsymbol{\psi}(k))$. Let us define

$$I^k := \{i |\, \psi_i(k) > 0\}.$$

Then, by Corollary 1 we have each $z_i(k+1)$, $i \in I^{k+1}$, is a convex combination of $z_j(k)$'s for $j \in I^k$. Therefore,

$$\begin{aligned} \max_{i \in I^{k+1}} z_i(k+1) &\leq \max_{i \in I^k} z_i(k), \\ \min_{i \in I^{k+1}} z_i(k+1) &\geq \min_{i \in I^k} z_i(k). \end{aligned} \tag{2.56}$$

These equations will be key to the analysis of the algorithm. We stress that we have not shown that the quantity $\min_i z_i(k)$ is non-decreasing; rather, we have shown that the related quantity, where the minimum is taken over I^k, the set of nonzero entries of $\boldsymbol{\psi}(k)$, is non-increasing.

Our next lemma provides lower and upper bounds on the entries of the vector $\boldsymbol{\psi}(k)$.

Lemma 18. *For $k \geq 0$ and $1 \leq i \leq n$ we have:*

$$n\alpha \leq \psi_i(k) \leq n.$$

Moreover, for $n+1 \leq h \leq n+m'$ and $k \geq 1$ we have either $\psi_h(k) = 0$ or,

$$n\alpha^2 \leq \psi_h(k) \leq n.$$

Proof. We have,

$$\boldsymbol{\psi}(k) = \mathbf{M}^{k-1:0} \begin{bmatrix} \mathbf{1}_n \\ \mathbf{0}_{m'} \end{bmatrix},$$

If $k < n\Gamma_s$, positive entries of $\mathbf{M}^{k-1:0}$ are at least $(1/n)^k$. Hence, positive entries of $\boldsymbol{\psi}(k)$ are at least,

$$\left(\frac{1}{n}\right)^k \geq \left(\frac{1}{n}\right)^{n\Gamma_s - 1} = n\alpha.$$

Now suppose $k \geq n\Gamma_s$. $\mathbf{M}^{k-1:0}$ is the product of $\mathbf{M}^{k-1:k-n\Gamma_s}$ and another column stochastic matrix. By Lemma 15, $\mathbf{M}^{k-1:k-n\Gamma_s}$ has positive first n rows, and positive entries of at least α. Thus, $\mathbf{M}^{k-1:0}$ has positive first n rows, and positive entries of at least α as well. We obtain for $1 \leq i \leq n$,

$$\psi_i(k) \geq n\alpha, \text{ for } k \geq 1.$$

For $n+1 \leq h \leq n+m'$, suppose ψ_h corresponds to a virtual node a_h corresponding to some link $(i,j) \in \mathcal{E}$. If $\psi_h(k)$ is positive, it is carrying a value sent from i at $k - n\Gamma_s$ or later, which has experienced link failure or delays. This is because each value gets to its destination after at most Γ_s iterations. Since i has self-loops all the time, a_h is reachable from i in period $k - n\Gamma_s$ to $k-1$; Hence, $M_{hi}^{k-1:k-n\Gamma_s} \geq \alpha$, and it follows,

$$\psi_h(k) \geq \alpha\psi_i(k - n\Gamma_s) \geq n\alpha^2.$$

Also, due to sum preservation property, we have $\psi_h(k) \leq n$, for all h and $k \geq 0$. □

Using Lemma 16 again, it follows,

$$\mathbf{z}(k + n\Gamma_s) = \hat{\mathbf{P}}(k)\mathbf{z}(k),$$

where,

$$\hat{\mathbf{P}}(k) := \text{diag}(\boldsymbol{\psi}^-(k + n\Gamma_s))\mathbf{M}^{k+nLs-1:k}\text{diag}(\boldsymbol{\psi}(k)). \tag{2.57}$$

Next, we are able to find a lower bound on the positive elements of $\hat{\mathbf{P}}(k)$. The proof of the following corollary is immediate.

Corrollary 2. *By* (2.57) *and Lemma 18 we have:*

(a) $\hat{P}_{ij}(k) > 0$ *for* $1 \leq i, j \leq n$.

(b) Positive entries of first n *columns of* $\hat{P}(k)$ *are at least* $(1/n)\alpha(n\alpha) = \alpha^2$. *Similarly, the last* m' *columns have positive entries of at least* α^3.

(c) For $h > n$, *if* $h \in I^{k+n\Gamma_s}$ *then* $\hat{P}_{hi}(k) > 0$ *for some* $1 \leq i \leq n$.

Our next lemma, which is the final result we need before proving the exponential convergence rate of RAPS, provides a quantitative bound for how multiplication by the matrix $\mathbf{P}$ shrinks the range of a vector.

Lemma 19. *Let* $t \geq 0$ *and* $\{\mathbf{u}(k)\}_{k\geq 0} \in \mathbb{R}^{n+m'}$ *be a sequence of vectors such that,*

$$\mathbf{u}(k+1) = \hat{\mathbf{P}}(kn\Gamma_s + t)\mathbf{u}(k).$$

Define

$$s_t(k) := \max_{i \in I^{kn\Gamma_s + t}} u_i(k) - \min_{i \in I^{kn\Gamma_s + t}} u_i(k).$$

Then,

$$s_t(k+2) \leq (1 - n\alpha^6)s_t(k).$$

Proof. Let us define

$$r_t(k) := \max_{1 \leq i \leq n} u_i(k) - \min_{1 \leq i \leq n} u_i(k).$$

By Corollary 2 for $j \in I^{(k+1)n\Gamma_s + t}$, the jth row of $\hat{\mathbf{P}}(kn\Gamma_s + t)$ has at least one positive entry in the first n columns. Thus, because $u_j(k+1)$ is maximized/minimized when all of the weight is put on the largest/smallest possible entry of $u_j(k)$, we have:

$$u_j(k+1) \leq \alpha^3 \max_{1 \leq i \leq n} u_i(k) + (1 - \alpha^3) \max_{i \in I^{kn\Gamma_s + t}} u_i(k),$$
$$u_j(k+1) \geq \alpha^3 \min_{1 \leq i \leq n} u_i(k) + (1 - \alpha^3) \min_{i \in I^{kn\Gamma_s + t}} u_i(k),$$

Therefore,

$$s_t(k+1) \leq \alpha^3 r_t(k) + (1 - \alpha^3)s_t(k). \tag{2.58}$$

Moreover, by a similar argument for $j \leq n$,

$$u_j(k+1) \leq \alpha^3 \sum_{i=1}^{n} u_i(k) + (1 - n\alpha^3) \max_{i \in I^{kn\Gamma_s + t}} u_i(k),$$
$$u_j(k+1) \geq \alpha^3 \sum_{i=1}^{n} u_i(k) + (1 - n\alpha^3) \min_{i \in I^{kn\Gamma_s + t}} u_i(k).$$

Thus,

$$r_t(k+1) \leq (1 - n\alpha^3) s_t(k).$$

Combining with (2.58) and noting that $r_t(k) \leq s_t(k)$ and $s_t(k+1) \leq s_t(k)$ we obtain,

$$\begin{aligned} s_t(k+2) &\leq \alpha^3(1 - n\alpha^3) s_t(k) + (1 - \alpha^3) s_t(k+1) \\ &\leq \alpha^3(1 - n\alpha^3) s_t(k) + (1 - \alpha^3) s_t(k) \\ &= (1 - n\alpha^6) s_t(k). \end{aligned}$$

□

Proof. **of Theorem 2** Using Lemma 19 with $t = 0$ and $\mathbf{u}(k) = \mathbf{z}(kn\Gamma_s)$ we get $s_0(k) \leq (1 - n\alpha^6)^{\lfloor k/2 \rfloor} s_0(0)$ and $\lim_{k \to \infty} s_0(k) = 0$. Moreover by (2.56), $\mathbf{z}_{\max}(k)$ is a non-increasing sequence and by $\mathbf{z}_{\min}(k)$, is non-decreasing. Thus,

$$\lim_{k \to \infty,\, h \in I^k} \mathbf{z}_h(k) = L_\infty. \tag{2.59}$$

We have:

$$\begin{aligned} L_\infty &= L_\infty \lim_{k \to \infty} \frac{\sum_{i=1}^{n+m'} \psi_i(k)}{\sum_{i=1}^{n+m'} \psi_i(k)} \\ &= \lim_{k \to \infty} \left(\frac{\sum_{i=1}^{n+m'} z_i(k)\psi_i(k)}{n} + \frac{\sum_{i=1}^{n+m'} (L_\infty - z_i(k))\psi_i(k)}{n} \right) \\ &= \lim_{k \to \infty} \left(\frac{\sum_{i=1}^{n+m'} \chi_i(k)}{n} + \frac{\sum_{i=1}^{n+m'} (L_\infty - z_i(k))\psi_i(k)}{n} \right) \\ &= \frac{\sum_{i=1}^{n} x_i(0)}{n}. \end{aligned}$$

In the above, we used (2.53) and (2.59), the boundedness of $\psi_i(k)$, and the fact that $\psi_i(k) = 0$ for $i \notin I^k$.

Finally, to show the exponential convergence rate, we go back to $s_0(k)$. We have for

$k \geq 1$,

$$s_0(k) \leq (1 - n\alpha^6)^{\lfloor k/2 \rfloor} s_0(0) \leq (1 - n\alpha^6)^{(k-1)/2} s_0(0),$$
$$s_0(0) \leq \sum_{i=1}^{n+m'} |z_i(0)| = \sum_{i=1}^{n} |x_i(0)| = \|\mathbf{x}(0)\|_1,$$

where the first equality holds because $I^0 = \{1, \ldots, n\}$ and $y_i(0) = 1$. Therefore, we have for $i \in I^k$,

$$\begin{aligned}
\left| z_i(k) - \frac{\mathbf{1}^\top \mathbf{x}(0)}{n} \right| &\leq z_{\max}(k) - z_{\min}(k) \\
&\leq s_0(\lfloor k/n\Gamma_s \rfloor) \\
&\leq (1 - n\alpha^6)^{(\lfloor \frac{k}{n\Gamma_s} \rfloor - 1)/2} \|\mathbf{x}(0)\|_1 \\
&\leq (1 - n\alpha^6)^{(\frac{k}{n\Gamma_s} - 1 - 1)/2} \|\mathbf{x}(0)\|_1 \\
&= \frac{1}{1 - n\alpha^6} \left((1 - n\alpha^6)^{1/(2n\Gamma_s)} \right)^k \|\mathbf{x}(0)\|_1 \\
&= \delta \lambda^k \|\mathbf{x}(0)\|_1.
\end{aligned}$$

where $\delta = \frac{1}{1-n\alpha^6}$ and $\lambda = (1 - n\alpha^6)^{1/(2n\Gamma_s)}$. Note that $\{1, \ldots, n\} \subseteq I^k$, $\forall k$. □

Remark: Observe that our proof did not really use the initialization $\boldsymbol{\psi}(0) = \mathbf{1}$, except to observe that the elements $\boldsymbol{\psi}(0)$ are positive, add up to n, and the implication that $\boldsymbol{\psi}(k)$ satisfies the bounds of Lemma 18. In particular, the same result would hold if we viewed time 1 as the initial point of the algorithm (so that $\boldsymbol{\psi}(1)$ is the initialization), or similarly any time k. We will use this observation in the next subsection.

2.5.3 Perturbed push-sum

In this subsection, we begin by introducing the Perturbed Robust Asynchronous Push-Sum algorithm, obtained by adding a perturbation to the x-values of (non-virtual) agents at the beginning of every iteration they wake up.

Algorithm 2 Perturbed Robust Asynchronous Push-Sum

1: Initialize the algorithm with $\mathbf{y}(0) = \mathbf{1}$, $\phi_i(0) = 0$, $\forall i \in \{1, \ldots, n\}$ and $\rho_{ij}(0) = 0$, $\kappa_{ij} = 0$, $\forall (j,i) \in \mathcal{E}$ and $\boldsymbol{\Delta}^0 = \mathbf{0}$.
2: At every iteration $k = 0, 1, 2, \ldots$, for every node i:
3: **if** node i wakes up **then**
4: $\quad x_i \leftarrow x_i + \Delta_i^k$;
5: $\quad$ Lines 4 to 17 of Algorithm 1
6: **end if**
7: Other variables remain unchanged.

Theorem 3. *Suppose Assumption 2 holds. Consider the sequence $\{z_i(k)\}$, $1 \le i \le n$, generated by Algorithm 2. Then,*

(a) For $k = 1, 2, \ldots$

$$\left| z_i(k) - \frac{\mathbf{1}^\top \boldsymbol{\chi}(k)}{n} \right| \le \delta \lambda^k \|\mathbf{x}^0\|_1 + \sum_{t=1}^{k-1} \delta \lambda^{k-t} \|\boldsymbol{\Delta}^t\|_1.$$

(b) If $\lim_{t\to\infty} \|\boldsymbol{\Delta}^t\|_1 = 0$ then,

$$\lim_{k\to\infty} \left| z_i(k) - \frac{\mathbf{1}^\top \boldsymbol{\chi}(k)}{n} \right| = 0.$$

where $\boldsymbol{\chi}(k)$ is an augmented vector of $\mathbf{x}^k$ which tracks the "masses" that are to be received by agents. It holds $\mathbf{1}^\top \boldsymbol{\chi}(k) = \mathbf{1}^\top \mathbf{x}^0 + \sum_{t=1}^{k-1} \mathbf{1}^\top \boldsymbol{\Delta}^t$

We show that, if the perturbations are bounded, the resulting $\mathbf{z}(k)$ nevertheless tracks the average of $\boldsymbol{\chi}(k)$ pretty well. Such a result is a key step towards analyzing distributed optimization protocols. In this general approach to the analyses of distributed optimization methods, we follow Ram et al. [2010] where it was first adopted; see also Nedić and Olshevsky [2016] and Nedić and Olshevsky [2015] where it was used.

Proof. Adopting the notations introduced earlier and by the linear formulation (2.52) we have,

$$\boldsymbol{\chi}(k+1) = \mathbf{M}(k)(\boldsymbol{\chi}(k) + \boldsymbol{\Delta}(k)), \qquad \text{for } k \ge 0,$$

where $\boldsymbol{\Delta}(k) \in \mathbb{R}^{n+m'}$ collects all perturbations $\Delta_i(k)$ in a column vector with $\Delta_h(k) := 0$

for $n < h \leq n + m'$. We may write this in a convenient form as follows.

$$\begin{aligned}\boldsymbol{\chi}(k+1) &= \mathbf{M}(k)(\boldsymbol{\chi}(k) + \boldsymbol{\Delta}(k)) \\ &= \sum_{t=1}^{k} \mathbf{M}^{k:t}\boldsymbol{\Delta}(t) + \mathbf{M}^{k:0}\boldsymbol{\chi}(0).\end{aligned}$$

Define for $k \geq 1$,

$$\begin{aligned}\boldsymbol{\chi}^t(k) &:= \mathbf{M}^{k-1:t}\boldsymbol{\Delta}(t), \qquad 1 \leq t \leq k, \\ \boldsymbol{\chi}^0(k) &:= \mathbf{M}^{k-1:0}\boldsymbol{\chi}(0), \qquad t = 0.\end{aligned} \tag{2.60}$$

We obtain,

$$\boldsymbol{\chi}(k) = \sum_{t=0}^{k-1} \boldsymbol{\chi}^t(k), \qquad k \geq 1. \tag{2.61}$$

Define $\mathbf{z}^t(k) := \boldsymbol{\chi}^t(k) \circ \boldsymbol{\psi}^-(k)$ for $0 \leq t \leq k$ (cf. (2.55)). We have

$$\mathbf{z}(k) = \sum_{t=0}^{k-1} \mathbf{z}^t(k). \tag{2.62}$$

We may view each $\mathbf{z}^t(k)$ as the outcome of a push-sum algorithm, initialized at time t, and apply Theorem 2. This immediately yields the following result, with part (b) an immediate consequence of part (a). □

2.6 Conclusion

In this chapter we established sufficient conditions on connectivity and link failures for consensus algorithms to converge. We started by showing that ordinary consensus and push-sum still work if intercommunication intervals do not grow too fast. Then we moved on to our main result, which is a fully asynchronous push-sum algorithm robust to link failures (RAPS). We proved its convergence while allowing consecutive link failures to grow to infinity, as long as they remain smaller than a logarithmically growing upper bound. We further analyzed RAPS under bounded delays, link failures and asynchrony and proved its geometrically convergence to average. Moreover, we provided convergence guarantees when the iterates are perturbed.

Chapter 3

Robust Asynchronous Stochastic Gradient-Push: Asymptotically Optimal and Network-Independent Performance for Strongly Convex Functions

3.1 Introduction

Considers the separable optimization problem

$$\min_{\mathbf{z}\in\mathbb{R}^d} F(\mathbf{z}) := \sum_{i=1}^{n} f_i(\mathbf{z}), \tag{3.1}$$

where the function $f_i : \mathbb{R}^d \to \mathbb{R}$ is held only by agent i in the network. We assume the agents communicate through a directed communication network, with each agent able to send messages to its out-neighbors. The agents seek to collaboratively agree on a minimizer to the global function $F(\mathbf{z})$.

This fairly simple problem formulation is capable of capturing a variety of scenarios in estimation and learning. Informally, $\mathbf{z}$ is often taken to parameterize a model, and $f_i(\mathbf{z})$ is a loss function measuring how well $\mathbf{z}$ matches the data held by agent i. Agreeing on a minimizer of $F(\mathbf{z})$ means agreeing on a model that best explains all the data throughout the network – and the challenge is to do this in a distributed manner, avoiding techniques such as flooding which requires every node to learn and store all the data throughout the network. For more details, we refer the reader to the recent survey by Nedić et al. [2018].

In this work, we will consider a fairly harsh network environment, including message losses, delays, asynchronous updates, and directed communication. The function $F(\mathbf{z})$ will be assumed to be strongly convex with the individual functions $f_i(\mathbf{z})$ having a Lipschitz continuous gradient. We will also assume that, at every time step, node i can obtain a

noisy gradient of its function $f_i(\mathbf{z})$. Our goal will be to investigate to what extent distributed methods can remain competitive with their centralized counterparts in spite of these obstacles.

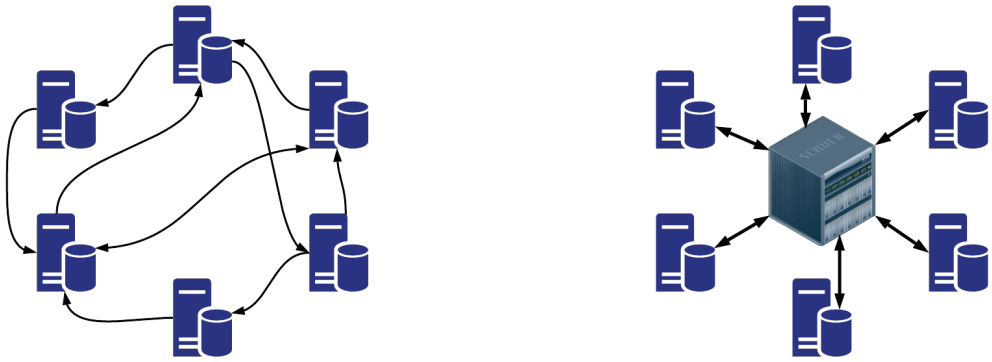
3.1.1 Literature review

Research on models of distributed optimization dates back to the 1980s, see Tsitsiklis et al. [1986]. The separable model of (3.1) was first formally analyzed in Nedić and Ozdaglar [2009], where performance guarantees on a fixed stepsize subgradient method were obtained. The literature on the subject has exploded since, and we review here only the papers closely related to our work. We begin by discussing works that have focused on the effect of harsh network conditions.

Asynchronous algorithms are often preferred to synchronous ones, due to the difficulty of perfectly coordinating all the agents in the network, e.g., due to clock drift. A number of recent papers have studied asynchronicity in the context of distributed optimization. Papers by Recht et al. [2011], Li et al. [2014], Agarwal and Duchi [2011], Lian et al. [2015] and Feyzmahdavian et al. [2016] study asynchronous parallel optimization methods in which different processors have access to a shared memory or parameter server. Recht et al. [2011] present a scheme called HOGWILD!, in which processors have access to the same shared memory with the possibility of overwriting each other's work. Li et al. [2014] proposes a parameter server framework for distributed machine learning. Agarwal and Duchi [2011] analyze the convergence of gradient-based optimization algorithms whose updates depend on delayed stochastic gradient information due to asynchrony. Lian et al. [2015] improve on the earlier work by Agarwal and Duchi [2011], and study two asynchronous parallel implementations of Stochastic Gradient (SG) for nonconvex optimization; establishing an $\mathcal{O}(1/\sqrt{k})$ convergence rate for both algorithms. Feyzmahdavian et al. [2016] propose an asynchronous mini-batch algorithm that eliminates idle waiting and allows workers to run at their maximal update rates.

The works mentioned above consider a *centralized* network topology, i.e., there is a

central node (parameter server or shared memory) connected to all the other nodes. On the other hand, in a *decentralized* setting, nodes communicate with each other over a connected network without depending on a central node (see Figure 3·1). This setting reduces the communication load on the central node, is not vulnerable to failures of that node, and is more easily scalable.



(a) Decentralized network topology. (b) Centralized network topology.

Figure 3·1: Different network topologies.

For analysis of how decentralized asynchronous methods perform we refer the reader to Mansoori and Wei [2017], Tsitsiklis et al. [1986], Srivastava and Nedić [2011], Assran and Rabbat [2018], Nedić [2011], Wu et al. [2018] and Tian et al. [2018]. We note that of these works only Tian et al. [2018] is able to obtain an algorithm which agrees on a global minimizer of (3.1) with non-random asynchronicity, under the assumptions of strong convexity, noiseless gradients and possible delays. On the other hand, the papers Nedić [2011] and Wu et al. [2018] obtain convergence in this situation under assumptions of natural randomness in the algorithm: the former assumes randomly failing links while the latter assumes that nodes make updates in random order.

The study of distributed separable optimization over directed graphs was initiated in Tsianos et al. [2012b], where a distributed approach based on dual averaging with convex functions over a fixed graph was proposed and shown to converge at an $\mathcal{O}(1/\sqrt{k})$ rate. Some numerical results for such methods were reported in Tsianos et al. [2012a]. In Nedić

and Olshevsky [2015], a method based on plain gradient descent converging at a rate of $\mathcal{O}((\ln k)/\sqrt{k})$ was proposed over time-varying graphs. This was improved in Nedić and Olshevsky [2016] to $\mathcal{O}((\ln k)/k)$ for strongly convex functions with noisy gradient samples. One of the recent works on optimization over *directed* graphs is Akbari et al. [2017], which considered online convex optimization in this setting, and Assran and Rabbat [2018], which considered combining directed graphs with delays and asynchronicity. The main tool for distributed optimization is the so-called "push sum" method introduced in Kempe et al. [2003], which is widely used to design communication and optimization schemes over directed graphs. More recent references are Bénézit et al. [2010], Hadjicostis et al. [2016], which provide a more modern and general analysis of this method, and the most comprehensive reference on the subject is the recent monograph by Hadjicostis et al. [2018]. We also mention Xi and Khan [2017a], Xi et al. [2018], Nedić et al. [2017], where an approach based on push-sum was explored. A parallel line of work in this setting based on the the ADMM model, where updates are allowed to include a local minimization step, was explored in Brisimi et al. [2018], Chang et al. [2016a,b] and Hong [2017].

The reason directed graphs present a problem is because much of distributed optimization relies on the primitive of "multiplication by a doubly stochastic matrix:" given that each node of a network holds a number x_i, the network needs to compute y_i, where $\mathbf{x} = (x_1, \ldots, x_n)^\top$, $\mathbf{y} = (y_1, \ldots, y_n)^\top$ and $\mathbf{y} = \mathbf{W}\mathbf{x}$ for some doubly stochastic matrix $\mathbf{W}$ with positive spectral gap. This is pretty easy to accomplish over undirected graphs (see Nedić et al. [2018]) but not immediate over directed graphs. A parallel line of research focuses on distributed methods for constructing such doubly stochastic matrices over directed graphs – we refer the reader to Dominguez-Garcia and Hadjicostis [2013], Gharesifard and Cortés [2012], Domínguez-García and Hadjicostis [2014]. Unfortunately, to the authors' best knowledge, no explicit and favorable convergence time guarantees are known for this procedure. Another line of work Xi and Khan [2017b] takes a similar approach, based on construction of a doubly stochastic matrix with positive spectral gap after the introduction of auxiliary states. Among works with undirected graphs, Scaman et al. [2017] derived the

optimal convergence rates for smooth and strongly convex functions and introduced the multi-step dual accelerated (MSDA) algorithm with optimal linear convergence rate in the deterministic case.

Dealing with message losses has always been a challenging problem for multi-agent optimization protocols. Recently, Hadjicostis et al. [2016] resolved this issue rather elegantly for the problem of distributed average computation by having nodes exchange certain running sums. It was shown in Hadjicostis et al. [2016] that the introduction of these running sums is equivalent to a lossless algorithm on a slightly modified graph. We also refer the reader to the follow-up papers by Su and Vaidya [2016b,a, 2017]. We will use the same approach in this work to deal with message losses.

In many applications, calculating the exact gradients can be computationally very expensive or impossible [Lan et al., 2020]. In one possible scenario, nodes are sensors that collect measurements at every step, which naturally corrupts all the data with noise. Alternatively, communication between agents may insert noise into information transmitted between them. Finally, when $f_i(\mathbf{z})$ measures the fit of a model parameterized by the vector $\mathbf{z}$ to the data of agent i, it may be efficient for agent i to randomly select a subset of its data and compute an estimate of the gradient based on only those data points [Alpcan and Bauckhage, 2009]. Motivated by these considerations, a literature has arisen studying the effects of stochasticity in the gradients. For example, Srivastava and Nedić [2011] showed convergence of an asynchronous algorithm for constrained distributed stochastic optimization, under the presence of local noisy communication in a random communication network. In Pu and Nedić [2018], two distributed stochastic gradient methods were introduced, and their convergence to a neighborhood of the global minimum (under constant step-size) and to the global minimum (under diminishing stepsize) was analyzed. In work by Sirb and Ye [2016], convergence of asynchronous decentralized optimization using delayed stochastic gradients has been shown.

The algorithms we will study here for stochastic gradient descent are based on the standard "consensus+gradient descent" framework: nodes will take steps in the direction

of their gradients and then "reconcile" these steps by moving in the directions of an average of their neighbors in the graph. We refer the reader to Nedić et al. [2018], Yuan et al. [2016], for a more recent and simplified analysis of such methods. It is also possible to take a more modern approach, pioneered in Shi et al. [2015], of using the past history to make updates; such schemes have been shown to achieve superior performance in recent years (see Shi et al. [2015], Sun et al. [2016], Oreshkin et al. [2010], Nedić et al. [2017], Xi and Khan [2017a], Xi et al. [2018], Qu and Li [2018], Xu et al. [2015], Qu and Li [2019], Di Lorenzo and Scutari [2016]); we refer the reader to Pu and Nedić [2018] which took this approach.

One of our main concerns in this chapter is to develop decentralized optimization methods which perform as well as their centralized counterparts. Specifically, we will compare the performance of a distributed method for (3.1) on a network of n nodes with the performance of a centralized method which, at every step, can query all n gradients of the functions $f_1(\mathbf{z}), \ldots, f_n(\mathbf{z})$. Since the distributed algorithm gets noise-corrupted gradients, so should the centralized method. Thus, the natural approach is to compare the distributed method to centralized gradient descent which moves in the direction of the sum of the gradients of $f_1(\mathbf{z}), \ldots, f_n(\mathbf{z})$. This method of comparison keeps the "computational power" of the two nodes identical.

Traditionally, the bounds derived on distributed methods were considerably worse than those derived for centralized methods. For example, the papers by Nedić and Olshevsky [2015, 2016] had bounds for distributed optimization over directed graphs that were worse than the comparable centralized method (in terms of rate of error decay) by a multiplicative factor that, in the worst case, could be as large as $n^{\mathcal{O}(n)}$. This is typical over directed graphs, though better results are possible over undirected graphs. For example, in Olshevsky [2017], in the model of noiseless, undelayed, synchronous communication over an undirected graph, a distributed subgradient method was proposed whose performance, relative to a centralized method with the same computational power, was worse by a multiplicative factor of n.

The breakthrough papers by Chen and Sayed [2015]; Pu and Garcia [2017], Morral et al. [2017], were the first to address this gap. These papers studied the model where gradients are

corrupted by noise, which we also consider in this chapter. Chen and Sayed [2015] examined the mean-squared stability and convergence of distributed strategies with fixed step-size over graphs and showed the same performance level as that of a centralized strategy, in the small step-size regime. In Pu and Garcia [2017] it was shown that, for a certain stochastic differential equation paralleling network gradient descent, the performance of centralized and distributed methods were comparable. In Morral et al. [2017], it was proved, for the first time, that distributed gradient descent with an appropriately chosen step-size, asymptotically performs similarly to a centralized method that takes steps in the direction of the sum of the noisy gradients, assuming iterates will remain bounded almost surely. This was the first analysis of a decentralized method for computing the *optimal* solution with performance bounds matching its centralized counterpart.

Both Pu and Garcia [2017] and Morral et al. [2017] were over fixed, undirected graphs with no message loss or delays or asynchronicity. As shown in the paper by Morral et al. [2012], this turns out to be a natural consequence of the analysis of those methods. Indeed, on a technical level, the advantage of working over undirected graphs is that they allow for easy distributed multiplication by doubly-stochastic matrices; it was shown in Morral et al. [2012] that if this property holds only in expectation – that is, if the network nodes can multiply by random stochastic matrices that are only doubly stochastic in expectation – distributed gradient descent will not perform comparably to its centralized counterpart.

In parallel to this work, and in order to reduce communication bottlenecks, Koloskova et al. [2019] propose a decentralized SGD with communication compression that can achieve the centralized baseline convergence rate, up to a constant factor. When the objective functions are smooth but not necessarily convex, Lian et al. [2017] show that Decentralized Parallel Stochastic Gradient Descent (D-PSGD) can asymptotically perform comparably to Centralized PSGD in total computational complexity. However, they argue that D-PSGD requires much less communication cost on the busiest node and hence, can outperform C-PSGD in certain communication regimes. Again, both Koloskova et al. [2019] and Lian et al. [2017] are over fixed undirected graphs, without delays, link failures or asynchronicity.

The follow-up work by Lian et al. [2018], extends the D-PSGD to the asynchronous case.

3.1.2 Our contribution

We propose an algorithm which we call *Robust Asynchronous Stochastic Gradient Push (RASGP)* for distributed optimization from noisy gradient samples *over directed graphs with message losses, delays, and asynchronous updates.* We will assume gradients are corrupted with additive noise represented by independent random variables, with bounded support, and with finite variance at node i denoted by σ_i^2. Our main result is that the RASGP performs as well as the best bounds on centralized gradient descent that moves in the direction of the sum of noisy gradients of $f_1(\mathbf{z}), \ldots, f_n(\mathbf{z})$. Our results also hold if the underlying graphs are time-varying as long as there are no message losses. We give a brief technical overview of this result next.

We will assume that each function $f_i(\mathbf{z})$ is μ_i-strongly convex with L_i-Lipschitz gradient, where $\sum_i \mu_i > 0$ and $L_i > 0$, $i = 1, \ldots, n$. The RASGP will have every node maintain an estimate of the optimal solution which will be updated from iteration to iteration; we will use $\mathbf{z}_i(k)$ to denote the value of this estimate held by node i at iteration k. We will show that, for each node $i = 1, \ldots, n$,

$$\mathbb{E}\left[\|\mathbf{z}_i(k) - \mathbf{z}^*\|_2^2\right] = \frac{\Gamma_u \sum_{i=1}^n \sigma_i^2}{k(\sum_{i=1}^n \mu_i)^2} + \mathcal{O}\left(\frac{1}{k^{1.5}}\right), \tag{3.2}$$

where $\mathbf{z}^* := \arg\min F(\mathbf{z})$ and Γ_u is the *degree of asynchronicity*, defined as the maximum number of iterations between two consecutive updates of any agent. The leading term matches the best bounds for (centralized) gradient descent that takes steps in the direction of the sum of the noisy gradients of $f_1(\mathbf{z}), \ldots, f_n(\mathbf{z})$, every k/Γ_u iterations (see Nemirovski et al. [2009], Rakhlin et al. [2012]). Asymptotically, the performance of the RASGP is network independent: indeed, the only effect of the network or the number of nodes is on the constant factor within the $\mathcal{O}\left(1/k^{1.5}\right)$ term above. The asymptotic scaling as $\mathcal{O}(1/k)$ is optimal in this setting [Rakhlin et al., 2012].

Consider the case when all the functions are identical, i.e., $f_1(\mathbf{z}) = \cdots = f_n(\mathbf{z})$, and

$\Gamma_u = 1$. In this case, letting $\mu = \mu_i$ and $\sigma = \sigma_i$, we have that for each $i = 1, \ldots, n$, (3.2) reduces to

$$\mathbb{E}\left[\|\mathbf{z}_i(k) - \mathbf{z}^*\|_2^2\right] = \frac{\sigma^2/n}{k\mu^2} + \mathcal{O}\left(\frac{1}{k^{1.5}}\right).$$

In other words, asymptotically we get the variance reduction of a centralized method that simply averages the n noisy gradients at each step.

The implication of this result is that one can get the benefit of having n independent processors computing noisy gradients in spite of all the usual problems associated with communications over a network (i.e., message losses, latency, asynchronous updates, one-way communication). Of course, the caveat is that one must wait sufficiently long for the asymptotic decay to "kick in," i.e., for the second term on the right-hand side of (3.2) to become negligible compared to the first. We leave the analysis of the size of this transient period to future work and note here that it *will* depend on the network and the number of nodes.[1]

The RASGP is a variation on the usual distributed gradient descent where nodes mix consensus steps with steps in the direction of their own gradient, combined with a new step-size trick to deal with asynchrony. It is presented as Algorithm 3 in Section 3.2. For a formal statement of the results presented above, we refer the reader to Theorem 4.

We briefly mention two caveats. The first is that implementation of the RASGP requires each node to use the quantity $\sum_{i=1}^n \mu_i/n$ in setting its local stepsize. This is not a problem in the setting when all functions are the same, but, otherwise, $\sum_{i=1}^n \mu_i/n$ is a global quantity not immediately available to each node. Assuming that node i knows μ_i, one possibility is to use average consensus to compute this quantity in a distributed manner before running the RASGP (for example using the algorithm described in Chapter 2). The second caveat is that, like all algorithms based on the push-sum method, the RASGP requires each node to know its out-degree in the communication graph.

[1]It goes without saying that no analysis of distributed optimization can be wholly independent of the network or the number of nodes. Indeed, in a network of n nodes, the diameter can be as large as $n-1$, which means that, in the worst case, no bounds on global performance can be obtained during the first $n-1$ steps of any algorithm.

3.1.3 Organization of this chapter

We conclude this Introduction with Section 3.1.4, which describes the basic notation we will use throughout the remainder of the chapter. Section 3.2 provides the RASGP algorithm for distributed optimization, and then states our main result, namely the asymptotically network-independent and optimal convergence rate. Results from numerical simulations of our algorithm to illustrate its performance are provided in Section 3.3, followed by conclusions in Section 3.4.

3.1.4 Notations and definitions

We assume there are n agents $\mathcal{V} = \{1, \dots, n\}$, communicating through a fixed directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{E}$ is the set of directed arcs. We assume $\mathcal{G}$ does not have self-loops and is strongly connected.

3.2 Robust Asynchronous Stochastic Gradient-Push (RASGP)

In this section we present the main contribution of this chapter, a distributed stochastic gradient method with asymptotically network-independent and optimal performance over directed graphs which is robust to asynchrony, delays, and link failures.

Recall that we are considering a network $\mathcal{G}$ of n agents whose goal is to cooperatively solve the following minimization problem

$$\text{minimize } F(\mathbf{z}) := \sum_{i=1}^{n} f_i(\mathbf{z}), \qquad \text{over } \mathbf{z} \in \mathbb{R}^d,$$

where each $f_i : \mathbb{R}^d \to \mathbb{R}$ is a strongly convex function only known to agent i. We assume agent i has the ability to obtain noisy gradients of the function f_i.

The RASGP algorithm is given as Algorithm 3. Note that we use the notation $\hat{\mathbf{g}}_i(k)$ for a noisy gradient of the function $f_i(\mathbf{z})$ at $\mathbf{z}_i(k)$ i.e.,

$$\hat{\mathbf{g}}_i(k) = \mathbf{g}_i(k) + \boldsymbol{\varepsilon}_i,$$

where $\mathbf{g}_i(k) := \nabla f_i(\mathbf{z}_i(k))$ and $\boldsymbol{\varepsilon}_i$ is a random vector.

Algorithm 3 Robust Asynchronous Stochastic Gradient-Push (RASGP)

Initialize the algorithm with $\mathbf{y}(0) = \mathbf{1}$, $\boldsymbol{\phi}_i^x(0) = \mathbf{0}$, $\phi_i^y(0) = 0$, $\kappa_i(0) = -1$, $\forall i \in \{1, \ldots, n\}$ and $\boldsymbol{\rho}_{ij}^x(0) = \mathbf{0}$, $\rho_{ij}^y(0) = 0$, $\kappa_{ij}(0) = -1$, $\forall (j,i) \in \mathcal{E}$.
At every iteration $k = 0, 1, 2, \ldots$, for every node i:
if node i wakes up **then**
  $\beta_i(k) = \sum_{t=\kappa_i+1}^{k} \alpha(t)$;
  $\mathbf{x}_i \leftarrow \mathbf{x}_i - \beta_i(k)\hat{\mathbf{g}}_i(k)$;
  $\kappa_i \leftarrow k$;
  $\boldsymbol{\phi}_i^x \leftarrow \boldsymbol{\phi}_i^x + \frac{\mathbf{x}_i}{d_i^+ + 1}$, $\phi_i^y \leftarrow \phi_i^y + \frac{y_i}{d_i^+ + 1}$;
  $\mathbf{x}_i \leftarrow \frac{\mathbf{x}_i}{d_i^+ + 1}$, $y_i \leftarrow \frac{y_i}{d_i^+ + 1}$;
  Node i broadcasts $(\boldsymbol{\phi}_i^x, \phi_i^y, \kappa_i)$ to its out-neighbors: N_i^+
  Processing the received messages
  for $(\boldsymbol{\phi}_j^x, \phi_j^y, \kappa_j')$ in the inbox **do**
    if $\kappa_j' > \kappa_{ij}$ **then**
      $\boldsymbol{\rho}_{ij}^{*x} \leftarrow \boldsymbol{\phi}_j^x$, $\rho_{ij}^{*y} \leftarrow \phi_j^y$;
      $\kappa_{ij} \leftarrow \kappa_j'$;
    end if
  end for
  $\mathbf{x}_i \leftarrow \mathbf{x}_i + \sum_{j \in N_i^-} \left(\boldsymbol{\rho}_{ij}^{*x} - \boldsymbol{\rho}_{ij}^x\right)$, $y_i \leftarrow y_i + \sum_{j \in N_i^-} \left(\rho_{ij}^{*y} - \rho_{ij}^y\right)$;
  $\boldsymbol{\rho}_{ij}^x \leftarrow \boldsymbol{\rho}_{ij}^{*x}$, $\rho_{ij}^y \leftarrow \rho_{ij}^{*y}$;
  $\mathbf{z}_i \leftarrow \frac{\mathbf{x}_i}{y_i}$;
end if
Other variables remain unchanged.

The RASGP is based on a standard idea of mixing consensus and gradient steps, first analyzed in Nedić and Ozdaglar [2009]. The push-sum scheme of Chapter 2, inspired by Hadjicostis et al. [2016], is used instead of the consensus scheme, which allows us to handle delays, asynchronicity, and message losses; this is similar to the approach taken in Nedić and Olshevsky [2015]. We note that a new step-size strategy is used to handle asynchronicity: when a node wakes up, it takes steps with a step-size proportional to the sum of all the step-sizes during the period it slept. As far as we are aware, this idea is new.

We will be making the following assumption on the noise vectors.

Assumption 3. *$\boldsymbol{\varepsilon}_i$ is an independent random vector with bounded support, i.e., $\|\boldsymbol{\varepsilon}_i\| \leq b_i$, $i = 1, \ldots, n$. Moreover, $\mathbb{E}[\boldsymbol{\varepsilon}_i] = \mathbf{0}$ and $\mathbb{E}[\|\boldsymbol{\varepsilon}_i\|^2] \leq \sigma_i^2$.*

Next, we state and prove the main result of this chapter, which states the linear convergence rate of Algorithm 3.

Theorem 4. *Suppose that:*

1. *Assumptions 2 and 3 hold.*
2. *Each objective function $f_i(\mathbf{z})$ is μ_i-strongly convex over $\mathbb{R}^d$.*
3. *The gradients of each $f_i(\mathbf{z})$ are L_i-Lipschitz continuous, i.e., for all $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^d$,*

$$\|\mathbf{g}_i(\mathbf{z}_1) - \mathbf{g}_i(\mathbf{z}_2)\| \leq L_i \|\mathbf{z}_1 - \mathbf{z}_2\|.$$

Then, the RASGP algorithm with the step-size $\alpha(k) = n/(\mu k)$ for $k \geq 1$ and $\alpha(0) = 0$, will converge to the unique optimum $\mathbf{z}^*$ with the following asymptotic rate: for all $i = 1, \ldots, n$, we have

$$\mathbb{E}\left[\|\mathbf{z}_i(k) - \mathbf{z}^*\|^2\right] \leq \frac{\Gamma_u \sigma^2}{k\mu^2} + \mathcal{O}\left(\frac{1}{k^{1.5}}\right),$$

where $\sigma^2 := \sum_i \sigma_i^2$, $\mu = \sum_i \mu_i$.

Remark 1. *We note that each agent stores variables $\mathbf{x}_i, y_i, \kappa_i, \mathbf{z}_i, \boldsymbol{\phi}_i^x, \phi_i^y$ and $\boldsymbol{\rho}_{ij}^x, \rho_{ij}^y, \kappa_{ij}$ for all in-neighbors $j \in N_i^-$. Hence, the memory requirement of the RASGP algorithm for each agent is $\mathcal{O}(d_i^-)$ for each agent i.*

We next turn to the proof of Theorem 4. First, we observe that Algorithm 3 is a specific case of multi-dimensional Perturbed Robust Asynchronous Push-Sum. In other words, each coordinate of vectors $\mathbf{x}_i$, $\mathbf{z}_i$, $\boldsymbol{\phi}_i^x$ and $\boldsymbol{\rho}_{ij}^x$ will experience an instance of Algorithm 2. Hence, there exists an augmented graph sequence $\{\mathcal{H}(k)\}$ where the Algorithm 3 is equivalent to perturbed push-sum consensus on $\mathcal{H}(k)$ where each agent $a_h \in \mathcal{V}_A$ holds vectors $\mathbf{x}_h$ and y_h. In other words, we will be able to apply Theorem 3 to analyze Algorithm 3.

Our first step is to show how to decouple the action of Algorithm 3 coordinate by coordinate. For each coordinate $1 \leq \ell \leq d$, let $\boldsymbol{\chi}^\ell \in \mathbb{R}^{n+m'}$ stack up the ℓth entries of x-values of all agents (virtual and non-virtual) in $\mathcal{V}_A$. Additionally, define $\boldsymbol{\Delta}^\ell(k) \in \mathbb{R}^{n+m'}$

to be the vector stacking up the ℓth entries of perturbations. i.e.,

$$[\boldsymbol{\Delta}^{\ell}(k)]_i := \begin{cases} -\beta_i(k)[\hat{\mathbf{g}}_i(k)]_{\ell}, & \text{if } i \in \mathcal{V}, \tau_i(k) = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Then, by the definition of the algorithm, we have for all $\ell = 1, \ldots, d$,

$$\begin{aligned} \boldsymbol{\chi}^{\ell}(k+1) &= \mathbf{M}(k) \left(\boldsymbol{\chi}^{\ell}(k) + \boldsymbol{\Delta}^{\ell}(k) \right), \\ \boldsymbol{\psi}(k+1) &= \mathbf{M}(k) \boldsymbol{\psi}(k). \end{aligned} \tag{3.3}$$

These equations write out the action of Algorithm 3 on a coordinate-by-coordinate basis.

In order to prove Theorem 4, we need a few tools and lemmas. As already mentioned, our first step will be to argue that Algorithm 3 converges by application of Theorem 3. This requires showing the boundedness of the perturbations $\boldsymbol{\Delta}^{\ell}(k)$, which, as we will show, reduces to showing the vectors $\mathbf{z}_i(k)$ are bounded. The following lemma will be useful to establish this boundedness.

Lemma 20. *[Nedić and Olshevsky, 2016, Lemma 3] Let $q : \mathbb{R}^d \to \mathbb{R}$ be a ν-strongly convex function with $\nu > 0$ which has Lipschitz gradients with constant L. Let $\mathbf{v} \in \mathbb{R}^d$ and $\mathbf{u} \in \mathbb{R}^d$ defined by,*

$$\mathbf{u} = \mathbf{v} - \alpha(\nabla q(\mathbf{v}) + p(\mathbf{v})),$$

where $\alpha \in \left(0, \nu/8L^2\right]$ and $p : \mathbb{R}^d \to \mathbb{R}^d$ is a mapping such that,

$$\|p(\mathbf{v})\| \leq c, \qquad \text{for all } \mathbf{v} \in \mathbb{R}^d.$$

Then, there exists a compact set $\mathcal{S} \subset \mathbb{R}^d$ and a scalar R such that,

$$\|\mathbf{u}\| \leq \begin{cases} \|\mathbf{v}\|, & \text{for all } \mathbf{v} \notin \mathcal{S}, \\ R, & \text{for all } \mathbf{v} \in \mathcal{S}, \end{cases}$$

where,

$$\mathcal{S} := \{\mathbf{z} |\, q(\mathbf{z}) \leq q(\mathbf{0}) + 2\frac{\nu}{8L^2}\left(\|q(\mathbf{0})\|^2 + c^2\right)\} \cup B\left(\mathbf{0}, \frac{4c}{\nu}\right),$$

$$R := \max_{\mathbf{z} \in \mathcal{S}} \{\|\mathbf{z}\| + \frac{\nu}{8L^2}\|\nabla q(\mathbf{z})\|\} + \frac{\nu c}{8L^2}.$$

We now argue that the iterates generated by Algorithm 3 are bounded.

Lemma 21. *The iterates $\mathbf{z}_i(k)$ generated by Algorithm 3 will remain bounded.*

Proof. Let us adopt the notation $\boldsymbol{\psi}^-$ from previous sections and define $\mathbf{z}^\ell(k) := \boldsymbol{\chi}^\ell(k) \circ \boldsymbol{\psi}^-(k) \in \mathbb{R}^{n+m'}$. Moreover, adopt the notation $\mathbf{z}_h$ for virtual agent a_h, $h = n+1, \ldots, n+m'$, as $\mathbf{z}_h(k) := \mathbf{x}_h(k)/\psi_h(k)$. Also define $\mathbf{u}^\ell \in \mathbb{R}^{n+m'}$ by

$$\mathbf{u}^\ell(k) := \boldsymbol{\chi}^\ell(k) + \boldsymbol{\Delta}^\ell(k).$$

Since the perturbations are only added to the non-virtual agents, which have strictly positive y-values, we conclude $[u^\ell(k)]_h = 0$ if $\psi_h(k) = 0$. Hence, the assumptions of Lemma 16 and Corollary 1 are satisfied. Adopting the definition of I^k and $\mathbf{P}(k)$ from previous sections, we get for $i \in I^{k+1}$,

$$[z^\ell(k+1)]_i = \sum_{j \in I^k} P_{ij}(k) \frac{[u^\ell(k)]_j}{\psi_j(k)}.$$

Combining the equation above for $\ell = 1, \ldots, d$ we obtain:

$$\mathbf{z}_i(k+1) = \sum_{j \in I^k} P_{ij}(k) \frac{\mathbf{u}_j(k)}{\psi_j(k)}, \tag{3.4}$$

where $\mathbf{u}_j(k) \in \mathbb{R}^d$ is created by collecting the jth entries of all $\mathbf{u}^\ell(k)$, i.e.,

$$\mathbf{u}_i(k) = \begin{cases} \mathbf{x}_i(k) - \beta_i(k)\hat{\mathbf{g}}_i(k), & \text{if } i \in \mathcal{V} \text{ and } \tau_i(k) = 1, \\ \mathbf{x}_i(k), & \text{otherwise.} \end{cases}$$

Now consider each term on the right hand side of (3.4) for $j \in I^k$. Suppose $j \leq n$ and $\tau_j(k) = 1$, then we have:

$$\frac{\mathbf{u}_j(k)}{y_j(k)} = \mathbf{z}_j(k) - \frac{\beta_j(k)}{y_j(k)}(\nabla f_j(\mathbf{z}_j(k)) + \boldsymbol{\varepsilon}_j(k)).$$

Since $\lim_{k\to\infty} \alpha(k) = 0$ and $k - \kappa_i(k) \leq \Gamma_u$, $\lim_{k\to\infty} \beta_j(k) = 0$. Moreover, by Lemma 18, $y_j(k)$ is bounded below; thus, $\lim_{k\to\infty} \beta_j(k)/y_j(k) = 0$ and there exists k_j such that for $k \geq k_j$, $\beta_j(k)/y_j(k) \in \left(0, \mu_j/8L_j^2\right]$. Applying Lemma 20, it follows that for each j there exists a compact set $\mathcal{S}_j$ and a scalar R_j such that for $k \geq k_j$, if $\tau_j(k) = 1$,

$$\left\| \frac{\mathbf{u}_j(k)}{y_j(k)} \right\| \leq \begin{cases} \|\mathbf{z}_j(k)\|, & \text{if } \mathbf{z}_j(k) \notin \mathcal{S}_j, \\ R_j, & \text{if } \mathbf{z}_j(k) \in \mathcal{S}_j. \end{cases} \tag{3.5}$$

Moreover, if $\tau_j(k) = 0$ or $j > n$ we have,

$$\frac{\mathbf{u}_j(k)}{y_j(k)} = \mathbf{z}_j(k). \tag{3.6}$$

Let $k_z := \max_i k_i$. Using mathematical induction, we will show that for all $k \geq k_z$:

$$\max_{i \in I^k} \|\mathbf{z}_i(k)\| \leq \bar{R}, \tag{3.7}$$

where $\bar{R} := \max\{\max_i R_i, \max_{j \in I^{k_z}} \|\mathbf{z}_j(k_z)\|\}$. Equation (3.7) holds for $k = k_z$. Suppose it is true for some $k \geq k_z$. Then by (3.5) and (3.6) we have,

$$\left\| \frac{\mathbf{u}_i(k)}{y_i(k)} \right\| \leq \max\{R_i, \|\mathbf{z}_i(k)\|\} \leq \bar{R}. \tag{3.8}$$

Also by (3.4), for $i \in I^{k+1}$, $\mathbf{z}_i(k+1)$ is a convex combination of $\mathbf{u}_j(k)/y_j(k)$'s, where $j \in I^k$. Hence,

$$\|\mathbf{z}_i(k+1)\| \leq \sum_{j \in I^k} P_{ij} \left\| \frac{\mathbf{u}_j(k)}{\psi_j(k)} \right\| \leq \bar{R}.$$

Define $B_z := \max\{\bar{R}, \max_{i \in I^k, k < k_z} \|\mathbf{z}_i(k)\|\}$ and we have $\|\mathbf{z}_i(k)\| \leq B_z$, $\forall k \geq 0$. □

We next explore a convenient way to rewrite Algorithm 3. Let us introduce the quantity $\mathbf{w}_i(k)$, which can be interpreted as the x-value of agent i, if it performed a gradient step at every iteration, even when asleep:

$$\mathbf{w}_i(k) := \begin{cases} \mathbf{x}_i(k) - \left(\sum_{t=\kappa_i(k)+1}^{k-1} \alpha(t) \right) \mathbf{g}_i(k), & \text{if } i \in \mathcal{V}, \\ \mathbf{x}_i(k), & \text{otherwise.} \end{cases} \tag{3.9}$$

Also, define $\mathbf{w}^\ell \in \mathbb{R}^{n+m'}$ by collecting the ℓth dimension of all $\mathbf{w}_i$'s and $\bar{\mathbf{w}}(k) := (\sum_{i=1}^{n+m'} \mathbf{w}_i(k))/n$. Moreover, define $\mathbf{g}^\ell \in \mathbb{R}^{n+m'}$ by collecting the ℓth value of gradients of all agents (0 for virtual agents), i.e.,

$$[\mathbf{g}^\ell(k)]_i = \begin{cases} [\mathbf{g}_i(k)]_\ell, & \text{if } i \in \mathcal{V}, \\ 0, & \text{otherwise.} \end{cases}$$

Additionally, define $\hat{\boldsymbol{\varepsilon}}_i(k) \in \mathbb{R}^d$ as the noise injected to the system at time k by agent i, i.e.,

$$\hat{\boldsymbol{\varepsilon}}_i(k) = \begin{cases} \beta_i(k)\boldsymbol{\varepsilon}_i(k), & \text{if } i \in \mathcal{V} \text{ and } \tau_i(k) = 1, \\ \mathbf{0}, & \text{otherwise,} \end{cases}$$

and $\hat{\boldsymbol{\varepsilon}}^\ell(k) \in \mathbb{R}^{n+m'}$ as the vector collecting the ℓth values of all $\hat{\boldsymbol{\varepsilon}}_i(k)$'s.

We then have the following lemma.

Lemma 22.

$$\mathbf{w}^\ell(k+1) = \mathbf{M}(k)\left(\mathbf{w}^\ell(k) - \alpha(k)\mathbf{g}^\ell(k) - \hat{\boldsymbol{\varepsilon}}^\ell\right). \tag{3.10}$$

Proof. We consider two cases:

- If $\tau_i(k) = 0$, then (3.10) reduces to $\mathbf{w}_i(k+1) = \mathbf{w}_i(k) - \alpha(k)\mathbf{g}_i(k)$; noting that, because node i did not update at time k we have that $\mathbf{g}_i(k) = \mathbf{g}_i(k+1)$ and this is the correct update.
- For all other nodes (i.e., for both virtual nodes and nodes with $\tau_i(k) = 1$), we have $[\mathbf{w}^\ell(k) - \alpha(k)\hat{\mathbf{g}}^\ell(k) - \hat{\boldsymbol{\varepsilon}}^\ell(k)]_i = [\boldsymbol{\chi}^\ell(k) + \Delta^\ell(k)]_i$ in (3.3). Since $\boldsymbol{\chi}^\ell(k+1) = \mathbf{M}(k)(\boldsymbol{\chi}^\ell(k) + \boldsymbol{\Delta}^\ell(k))$ and, using the definition of $\mathbf{w}_i(k)$, we have that for these nodes,

  $$\mathbf{w}_i(k+1) = \mathbf{x}_i(k+1);$$

  (3.3) implies the conclusion.

□

This lemma allows us to straightforwardly analyze how the average of $\mathbf{w}(k)$ evolves. Indeed, summing all the elements of (3.10) and dividing by n for each $\ell = 1, \ldots, d$ we obtain,

$$\begin{aligned} \bar{\mathbf{w}}(k+1) &= \bar{\mathbf{w}}(k) - \frac{\alpha(k)}{n}\sum_{i=1}^n \mathbf{g}_i(k) - \frac{1}{n}\sum_{i=1}^n \hat{\boldsymbol{\varepsilon}}_i(k) \\ &= \bar{\mathbf{w}}(k) - \frac{\alpha(k)}{n}\sum_{i=1}^n \nabla f_i(\bar{\mathbf{w}}(k)) - \frac{1}{n}\sum_{i=1}^n \hat{\boldsymbol{\varepsilon}}_i(k) - \frac{\alpha(k)}{n}\sum_{i=1}^n \left(\mathbf{g}_i(k) - \nabla f_i(\bar{\mathbf{w}}(k))\right). \end{aligned} \tag{3.11}$$

We next give a sequence of lemmas to the effect that all the quantities generated by the algorithm are close to each other over time. Define,

$$\bar{\mathbf{x}}(k) = \frac{1}{n} \sum_{a_h \in \mathcal{V}_A} \mathbf{x}_h(k).$$

where, recall, $\mathcal{V}_A$ is our notation for all the nodes in the augmented graph (i.e., including virtual nodes). Moreover, we will extend the definition of $\beta_i(k)$ from Line 4 of Algorithm 3 to *all* k via the same formula $\beta_i(k) := \sum_{t=\kappa_i(k)+1}^{k} \alpha(t)$. Our first lemma will show that each $\mathbf{z}_i(k)$ closely tracks $\bar{\mathbf{x}}(k)$.

Lemma 23. *Using Algorithm 3 with $\alpha(k) = n/(k\mu)$, under the assumptions of Theorem 4, we have for each i, $\|\mathbf{z}_i(k+1) - \bar{\mathbf{x}}(k+1)\| = \mathcal{O}(1/k)$.*

Proof. By Theorem 3(a) we have for each ℓ,

$$\left| [\mathbf{z}^\ell(k+1)]_i - \frac{\mathbf{1}^\top \boldsymbol{\chi}^\ell(k+1)}{n} \right| \leq \delta \lambda^k \|\boldsymbol{\chi}^\ell(0)\|_1 + \sum_{t=1}^{k} \delta \lambda^{k-t} \|\boldsymbol{\Delta}^\ell(t)\|_1.$$

Summing the above inequality for $\ell = 1, \ldots, d$ we obtain,

$$\|\mathbf{z}_i(k+1) - \bar{\mathbf{x}}(k+1)\|_1 \leq \sum_{j=1}^{n} \Big(\delta \lambda^k \|\mathbf{x}_j(0)\|_1 + \sum_{t=1}^{k} \delta \lambda^{k-t} \beta_i(t) \tau_i(t) \|\hat{\mathbf{g}}_j(t)\|_1 \Big).$$

Moreover,

$$\beta_i(k) = \sum_{t=\kappa_i(k)+1}^{k} \frac{n}{\mu t} \leq \frac{n}{\mu} \left(\frac{k - \kappa_i(k)}{\kappa_i(k) + 1} \right). \tag{3.12}$$

But $\kappa_i(k) < k \leq \kappa_i(k) + \Gamma_u$. Since $\Gamma_u \geq 1$, we obtain

$$k \leq (\kappa_i(k) + 1)\Gamma_u,$$

or,

$$\frac{1}{\kappa_i(k) + 1} \leq \frac{\Gamma_u}{k}.$$

Thus, from (3.12) we have,

$$\beta_i(k) \leq \frac{n\Gamma_u^2}{\mu k}. \tag{3.13}$$

Define

$$M_j := \max_{\|\mathbf{z}\| \le B_z} \|\mathbf{g}_j(\mathbf{z})\|_1, \tag{3.14}$$

and observe that M_j is finite by Lemma 21. Also $\tau_j(k) \le 1$. We obtain,

$$\|\mathbf{z}_i(k+1) - \bar{\mathbf{x}}(k+1)\|_1 \le \sum_{j=1}^{n} \Big(\delta\lambda^k \|\mathbf{x}_j(0)\|_1 + \sum_{t=1}^{k} \delta\lambda^{k-t} \frac{n\Gamma_u^2}{\mu t}(M_j + b_j) \Big).$$

Let RHS denote the right hand side of the relation above. We have,

$$\begin{aligned} RHS &= \sum_{j=1}^{n} \left(\delta\lambda^k \|\mathbf{x}_j(0)\|_1 + \frac{\delta n\Gamma_u^2}{\mu}(M_j + b_j) \Big(\sum_{t=1}^{\lfloor \frac{k}{2} \rfloor} \frac{\lambda^{k-t}}{t} + \sum_{t=\lfloor \frac{k}{2} \rfloor + 1}^{k} \frac{\lambda^{k-t}}{t} \Big) \right) \\ &\le \sum_{j=1}^{n} \left(\delta\lambda^k \|\mathbf{x}_j(0)\|_1 + \frac{\delta n\Gamma_u^2}{\mu}(M_j + b_j) \left(\frac{k}{2}\lambda^{\frac{k}{2}} + \frac{2}{(1-\lambda)k} \right) \right) = \mathcal{O}\left(\frac{1}{k}\right), \end{aligned}$$

where we used the following relations,

$$\sum_{t=1}^{\lfloor \frac{k}{2} \rfloor} \frac{\lambda^{k-t}}{t} \le \lfloor \frac{k}{2} \rfloor \lambda^{k - \lfloor \frac{k}{2} \rfloor} \le \frac{k}{2}\lambda^{\frac{k}{2}},$$

$$\sum_{t=\lfloor \frac{k}{2} \rfloor + 1}^{k} \frac{\lambda^{k-t}}{t} \le \sum_{t=0}^{\lceil \frac{k}{2} \rceil - 1} \frac{\lambda^t}{\lfloor \frac{k}{2} \rfloor + 1} \le \frac{2}{(1-\lambda)k}.$$

Finally, $\|\mathbf{v}\|_2 \le \|\mathbf{v}\|_1$ for all vectors $\mathbf{v}$, completes the proof. □

An immediate consequence of this lemma is that the quantities $\bar{\mathbf{x}}(k)$ and $\bar{\mathbf{w}}(k)$ are close to each other.

Lemma 24. *Using Algorithm 3 with $\alpha(k) = n/(k\mu)$, under the assumptions of Theorem 4, we have, $\|\bar{\mathbf{x}}(k) - \bar{\mathbf{w}}(k)\| = \mathcal{O}(1/k)$.*

Proof. By definition of $\bar{\mathbf{w}}$ we have,

$$\bar{\mathbf{x}}(k) - \bar{\mathbf{w}}(k) = \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{t=\kappa_i(k)+1}^{k-1} \alpha(t) \right) \mathbf{g}_i(k).$$

Using (3.13) we have,

$$\|\bar{\mathbf{x}}(k) - \bar{\mathbf{w}}(k)\| \le \frac{1}{n} \sum_{i=1}^{n} \beta_i(k) M_i \le \sum_{i=1}^{n} \frac{\Gamma_u^2 M_i}{n\mu k} = \mathcal{O}\left(\frac{1}{k}\right),$$

where M_i was defined through (3.14). □

We next remark on a couple of implications of the past series of lemmas.

Corrollary 3. *We have* $\|\mathbf{z}_i(k) - \bar{\mathbf{w}}(k)\| = \mathcal{O}(1/k)$.

Lemma 25. $\|\mathbf{g}_i(k) - \nabla f_i(\bar{\mathbf{w}}(k))\| = \mathcal{O}(1/k)$.

Proof. Since ∇f_i is L_i-Lipschitz, we have,

$$\|\mathbf{g}_i(k) - \nabla f_i(\bar{\mathbf{w}}(k))\| \leq L_i \|\mathbf{z}_i(k) - \bar{\mathbf{w}}(k)\|.$$

Using Corollary 3, the lemma is proved. □

We are now in a position to rewrite Algorithm 3 as a sort of perturbed gradient descent. Let us define,

$$\boldsymbol{\eta}(k) := \frac{1}{\mu k} \sum_{i=1}^{n} \left(\mathbf{g}_i(k) - \nabla f_i(\bar{\mathbf{w}}(k))\right).$$

By Lemma 25, $\boldsymbol{\eta}(k) = \mathcal{O}(1/k^2)$. Therefore, there exists B_η such that $\boldsymbol{\eta}(k) \leq B_\eta/k^2$ for all $k \geq 1$.

By (3.11) we have,

$$\bar{\mathbf{w}}(k+1) = \bar{\mathbf{w}}(k) - \frac{1}{\mu k} \nabla F(\bar{\mathbf{w}}(k)) - \bar{\boldsymbol{\varepsilon}}(k) - \boldsymbol{\eta}(k), \tag{3.15}$$

where

- The function $F := \sum_{i=1}^{n} f_i \in \mathbb{R}^d \to \mathbb{R}$ is μ-strongly-convex with L-Lipschitz gradient, where $L := \sum_{i=1}^{n} L_i$.
- The noise $\bar{\boldsymbol{\varepsilon}}(k) := (\sum_{i=1}^{n} \hat{\boldsymbol{\varepsilon}}_i(k))/n$ is bounded (i.e., $\bar{\boldsymbol{\varepsilon}}(k) \in B(0, r_e)$, with probability one, where $r_e := (\Gamma_u/\mu) \sum_j b_j$), and $\mathbb{E}[\bar{\boldsymbol{\varepsilon}}(k)] = \mathbf{0}$.

In other words, with the exception of the $\boldsymbol{\eta}(k)$ term, what we have is exactly a stochastic gradient descent method on the function $F(\cdot)$.

The following lemmas bound $\bar{\boldsymbol{\varepsilon}}(k)$. Let us define $\nu_i(k) = k - \kappa_i(k)$ as the number of iterations agent i has skipped since it's last update. By Assumption 2, $\nu_i(k) \leq \Gamma_u$.

Lemma 26. *We have $\beta_i(k) = \mathcal{O}(1/k)$, $\forall i$. Moreover,*

$$\beta_i(k) \leq \frac{n\nu_i(k)}{\mu k} + \mathcal{O}(k^{-2}).$$

Proof. Since $\nu_i(k) \leq \Gamma_u, \forall i$, we have for $\kappa_i(k) \geq 1$,

$$\begin{aligned}\beta_i(k) &= \sum_{t=\kappa_i(k)+1}^{k} \frac{n}{\mu t} \leq \frac{n}{\mu} \ln\left(\frac{k}{\kappa_i(k)}\right) \leq \frac{n}{\mu} \ln\left(\frac{k}{k-\nu_i(k)}\right) \\ &= \frac{n}{\mu} \ln\left(1 + \frac{\nu_i(k)}{k-\nu_i(k)}\right) \leq \frac{n\nu_i(k)}{\mu(k-\nu_i(k))} = \frac{n\nu_i(k)}{\mu k} + \mathcal{O}(k^{-2}).\end{aligned}$$

□

Corrollary 4. *$\mu k\|\bar{\boldsymbol{\varepsilon}}(k)\|$ is bounded.*

Lemma 27. *There exists $B_\epsilon > 0$ such that we have,*

$$\mathbb{E}[\|\bar{\boldsymbol{\varepsilon}}(k)\|^2] \leq \frac{\Gamma_u^2}{\mu^2 k^2}\sigma^2 + \frac{B_\epsilon}{k^4}.$$

Proof. Using Lemma 26, we have for $k > \Gamma_u$,

$$\begin{aligned}\mathbb{E}[\|\bar{\boldsymbol{\varepsilon}}(k)\|^2] &= \mathbb{E}[\|\frac{1}{n}\sum_{i=1}^{n} \beta_i(k)\boldsymbol{\varepsilon}_i(k)\tau_i(k)\|^2] = \frac{1}{n^2}\sum_{i=1}^{n} \beta_i^2(k)\mathbb{E}[\|\boldsymbol{\varepsilon}_i(k)\|^2] \\ &\leq \frac{1}{n^2}\sum_{i=1}^{n} \beta_i^2(k)\sigma_i^2 \leq \frac{\Gamma_u^2}{\mu^2 k^2}\sigma^2 + \mathcal{O}(k^{-4}),\end{aligned}$$

where the second equality is the result of the noise terms being independent and zero-mean. □

Our next observation is a technical lemma which is essentially a rephrasing of Lemma 20 above.

Lemma 28. *There exists a constant B_w and time k_w such that $\|\bar{\mathbf{w}}(k)\| \leq B_w$ with probability one, for $k \geq k_w$.*

Proof. We have

$$\bar{\mathbf{w}}(k+1) = \bar{\mathbf{w}}(k) - \frac{1}{\mu k}\left[\nabla F(\bar{\mathbf{w}}(k)) + \mu k\left(\bar{\boldsymbol{\varepsilon}}(k) + \boldsymbol{\eta}(k)\right)\right],$$

where $\mu k\|\bar{\boldsymbol{\varepsilon}}(k) + \boldsymbol{\eta}(k)\|$ is bounded. Moreover, there exists k_w such that for $k \geq k_w$, $\frac{1}{\mu k} \in \left(0, \mu/8L^2\right]$. Therefore, by Lemma 20 there exists a compact set $\mathcal{S}_w$ and a scalar

$R_w > 0$ such that for $k \geq k_w$,

$$\|\bar{\mathbf{w}}(k+1)\| \leq \begin{cases} \|\bar{\mathbf{w}}(k)\|, & \text{for } \bar{\mathbf{w}} \notin \mathcal{S}_w, \\ R_w, & \text{for } \bar{\mathbf{w}} \in \mathcal{S}_w. \end{cases}$$

Therefore, setting $B_w := \max\{R_w, \|\bar{\mathbf{w}}(k_w)\|\}$ will complete the proof. □

As a consequence of this lemma, because $\|\boldsymbol{\eta}(k)\|_2 \leq B_\eta$, this lemma implies there is a constant B_1 such that for $k \geq k_w$,

$$\left\| \bar{\mathbf{w}}(k) - \mathbf{z}^* - \frac{1}{\mu k} \nabla F(\bar{\mathbf{w}}(k)) - \bar{\boldsymbol{\varepsilon}}(k) \right\| \leq B_1, \tag{3.16}$$

with probability one. This now puts us in a position to show that $\bar{\mathbf{w}}(k)$ converges in mean square to the optimal solution.

Lemma 29. $\mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2] \to 0$.

Proof. Using the definition of k_w from Lemma 28, we have that for $k \geq k_w$,

$$\mathbb{E}[\|\bar{\mathbf{w}}(k+1) - \mathbf{z}^*\|^2] \leq \mathbb{E}\Big[\|\bar{\mathbf{w}}(k) - \mathbf{z}^* - \frac{1}{\mu k} \nabla F(\bar{\mathbf{w}}(k)) - \bar{\boldsymbol{\varepsilon}}(k)\|^2 + 2\|\boldsymbol{\eta}(k)\| \|\bar{\mathbf{w}}(k) - \mathbf{z}^* - \frac{1}{\mu k} \nabla F(\bar{\mathbf{w}}(k)) - \bar{\boldsymbol{\varepsilon}}(k)\| + \|\boldsymbol{\eta}(k)\|^2 \Big].$$

We will bound each of the terms on the right. We begin with the easiest one, which is the last one:

$$\|\boldsymbol{\eta}(k)\|^2 \leq \frac{B_\eta^2}{k^4}. \tag{3.17}$$

The middle term is bounded as

$$2\|\boldsymbol{\eta}(k)\| \|\bar{\mathbf{w}}(k) - \mathbf{z}^* - \frac{1}{\mu k} \nabla F(\bar{\mathbf{w}}(k)) - \bar{\boldsymbol{\varepsilon}}(k)\| \leq \frac{2 B_\eta B_1}{k^2}, \tag{3.18}$$

where we used (3.16).

Finally, we turn to the first term which we denote by T_1:

$$T_1 \leq \mathbb{E}\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2 - \frac{2}{\mu k} \mathbb{E}[\nabla F(\bar{\mathbf{w}}(k))^\top (\bar{\mathbf{w}}(k) - \mathbf{z}^*)] + \frac{L^2}{\mu^2 k^2} \mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2] + \mathbb{E}[\|\bar{\boldsymbol{\varepsilon}}(k)\|^2],$$

where we used the usual inequality $\|\nabla F(\bar{\mathbf{w}}(k))\|^2 \leq L^2\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2$ which follows from $\nabla F(\cdot)$ being L-Lipschitz. Now, using the standard inequality

$$\begin{aligned}\nabla F(\bar{\mathbf{w}}(k))^T(\bar{\mathbf{w}}(k) - \mathbf{z}^*) &\geq F(\bar{\mathbf{w}}(k)) - F(\mathbf{z}^*) + \frac{\mu}{2}\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2 \\ &\geq \mu\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2,\end{aligned}$$

and Lemma 27 we obtain,

$$T_1 \leq \left(1 - \frac{2}{k} + \frac{L^2}{\mu^2 k^2}\right)\mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2] + \frac{\Gamma_u^2}{\mu^2 k^2}\sigma^2 + \frac{B_\epsilon}{k^4}. \tag{3.19}$$

Now putting together (3.17), (3.18), and (3.19), we get,

$$\mathbb{E}[\|\bar{\mathbf{w}}(k+1) - \mathbf{z}^*\|^2] \leq \left(1 - \frac{2}{k} + \frac{L^2}{\mu^2 k^2}\right)\mathbb{E}[\|\bar{\mathbf{w}}(k) - x^*\|^2] + \frac{\Gamma_u^2\sigma^2}{\mu^2 k^2} + \frac{2B_\eta B_1}{k^2} + \frac{B_\eta^2 + B_\epsilon}{k^4}.$$

For large enough k, we can bound the inequality above as,

$$\mathbb{E}[\|\bar{\mathbf{w}}(k+1) - \mathbf{z}^*\|^2] \leq \left(1 - \frac{1.5}{k}\right)\mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2] + \frac{B_2}{k^2}, \tag{3.20}$$

where $B_2 = \Gamma_u^2\sigma^2/\mu^2 + 2B_\eta B_1 + B_\eta^2 + B_\epsilon$. Using Lemma 30, stated next, we conclude $\mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2] \to 0$. □

Lemma 30. *Let $a > 1, b \geq 0$ and $\{x_t\}$ be a non-negative sequence which satisfies,*

$$x_{t+1} \leq \left(1 - \frac{a}{t}\right)x_t + \frac{b}{t^2}, \qquad \text{for } t \geq t' > 0.$$

Then for all $t \geq t'$ we have,

$$x_t \leq \frac{m}{t},$$

where $m := \max\{t' x_{t'}, b/(a-1)\}$.

This lemma is stated and proved for $t' = 1$ in Rakhlin et al. [2012, Lemma 3], and the case of general t' follows immediately.

We are almost ready to complete the proof of Theorem 4; all that is needed is to refine the convergence rate of $\bar{\mathbf{w}}(k)$ to x^*. Now as a consequence of (3.20) and Lemma 30, we may use the inequality $\mathbb{E}[|X|] \leq \sqrt{\mathbb{E}[X^2]}$ to obtain that

$$\mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|] = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right). \tag{3.21}$$

Furthermore, by the finite support of $\mu k\bar{\boldsymbol{\varepsilon}}(k)$, by Corollary 4, we also have that

$$\mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^* - \frac{1}{\mu k}\nabla F(\bar{\mathbf{w}}(k)) - \bar{\boldsymbol{\varepsilon}}(k)\|] = \mathcal{O}\left(\frac{1}{\sqrt{k}}\right). \tag{3.22}$$

We now use these observations to provide a proof of our main result.

Proof of Theorem 4. Essentially, we rewrite the proof of Lemma 29, but now using the fact that $\mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|] = \mathcal{O}(1/\sqrt{k})$ from (3.21). This allows us to make two modification to the arguments of that lemma. First, we can now replace (3.18) by

$$\mathbb{E}[2\|\boldsymbol{\eta}(k)\|\|\bar{\mathbf{w}}(k) - \mathbf{z}^* - \frac{1}{\mu k}\nabla F(\bar{\mathbf{w}}(k)) - \bar{\boldsymbol{\varepsilon}}(k)\|] \leq \frac{2B_\eta}{k^2}\mathcal{O}\left(\frac{1}{\sqrt{k}}\right), \tag{3.23}$$

where we used (3.22). Second, putting together (3.17), (3.23), and (3.19), we obtain:

$$\mathbb{E}[\|\bar{\mathbf{w}}(k+1) - \mathbf{z}^*\|^2] \leq \left(1 - \frac{2}{k} + \frac{L^2}{\mu^2 k^2}\right)\mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2] \\ + \mathbb{E}[\|\bar{\boldsymbol{\varepsilon}}(k)\|^2] + \frac{B_\eta^2}{k^4} + \frac{2B_\eta}{k^2}\mathcal{O}\left(\frac{1}{\sqrt{k}}\right).$$

which, again using the fact that $\mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2] = \mathcal{O}(1/\sqrt{k})$, we simply rewrite as,

$$\mathbb{E}[\|\bar{\mathbf{w}}(k+1) - \mathbf{z}^*\|^2] \leq \left(1 - \frac{2}{k}\right)\mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2] + \mathbb{E}[\|\bar{\boldsymbol{\varepsilon}}(k)\|^2] + \mathcal{O}\left(\frac{1}{k^{2.5}}\right).$$

To save space, let us define $a_k := \mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2]$. Multiplying both sides of relation above by k^2 we obtain,

$$a_{k+1}k^2 \leq a_k\left(1 - \frac{2}{k}\right)k^2 + \mathbb{E}[\|\bar{\boldsymbol{\varepsilon}}(k)\|^2]k^2 + \mathcal{O}(k^{-0.5}).$$

Note that,

$$\left(1 - \frac{2}{k}\right)k^2 = k^2 - 2k < (k-1)^2.$$

Thus,

$$a_{k+1}k^2 \leq a_k(k-1)^2 + \mathbb{E}[\|\bar{\boldsymbol{\varepsilon}}(k)\|^2]k^2 + \mathcal{O}(k^{-0.5}).$$

Summing the relation above for $k = 0, \ldots, T$ implies,

$$a_{T+1}T^2 \leq \sum_{k=0}^{T}\mathbb{E}[\|\bar{\boldsymbol{\varepsilon}}(k)\|^2]k^2 + \mathcal{O}(T^{0.5}).$$

Now, let us estimate the first term on the right hand side of relation above,

$$\sum_{k=0}^{T}\mathbb{E}[\|\bar{\boldsymbol{\varepsilon}}(k)\|^2]k^2 \leq \sum_{k=0}^{T}\sum_{i=1}^{n}\frac{\beta_i^2(k)}{n^2}\sigma_i^2\tau_i(k)k^2 = \sum_{i=1}^{n}\frac{\sigma_i^2}{\mu^2}\sum_{k=0}^{T}\nu_i(k)^2\tau_i(k) + \mathcal{O}(T^{-1}),$$

where we used Lemma 26 in the last equality. Define $t_i(j)$ as the j'th time agent i has woken up, and set $t_i(0) = -1$. Then we can rewrite the relation above as,

$$\sum_{k=0}^{T}\nu_i(k)^2\tau_i(k) = \sum_{j=1}^{t_i(j)\leq T}(t_i(j) - t_i(j-1))^2 \leq \sum_{j=1}^{t_i(j)\leq T}\Gamma_u(t_i(j) - t_i(j-1)) \leq \Gamma_u(T+1).$$

Combining relations above and then dividing both sides by T^2 we obtain,

$$a_{T+1} \leq \frac{\Gamma_u\sigma^2}{\mu^2 T} + \mathcal{O}(T^{-1.5}). \tag{3.24}$$

We next argue that the same guarantee holds for every $\mathbf{z}_i(k)$. Indeed, for each $i = 1, \ldots, m$,

$$\begin{aligned}\|\mathbf{z}_i(k) - \mathbf{z}^*\|^2 &= \|\mathbf{z}_i(k) - \bar{\mathbf{w}}(k) + \bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2 \\ &= \|\mathbf{z}_i(k) - \bar{\mathbf{w}}(k)\|^2 + 2\|\mathbf{z}_i(k) - \bar{\mathbf{w}}(k)\|\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\| + \|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2.\end{aligned}$$

Now from Corollary 3, we know that with probability one, $\|\mathbf{z}_i(k) - \bar{\mathbf{w}}(k)\|_2 = \mathcal{O}(1/k)$. Taking expectation of both sides and using (3.24) along with the usual bound $\mathbb{E}[|X|] \leq \sqrt{\mathbb{E}[X^2]}$, we have

$$\mathbb{E}[\|\mathbf{z}_i(k) - \mathbf{z}^*\|^2] = \mathcal{O}\left(\frac{1}{k^2}\right) + \mathcal{O}\left(\frac{1}{k^{1.5}}\right) + \mathbb{E}[\|\bar{\mathbf{w}}(k) - \mathbf{z}^*\|^2].$$

Putting this together with (3.24) completes the proof. □

Time-varying graphs We remark that Theorems 2, 3 and 4 all extend verbatim to the case of time-varying graphs with no message losses. Indeed, only one problem appears in extending the proofs in this chapter to time-varying graphs: a node i may send a message to node j; that message will be lost; and afterwards node i never sends anything to node j again. In this case, some lemmas we proved do not hold. However, as long as no messages are lost, the proofs in this chapter extend to the time-varying case verbatim.

On the bounds for delays, asynchrony, and message losses It is natural to what extent the assumption of finite upper bounds on delays, asynchrony, and message losses are really necessary. A natural example which falls outside our framework is a fixed graph G, where, at each time step, every link in G appears with probability $1/2$. A more general model might involve a different probability p_e of failure for each edge e.

We observe that our result can already handle this case in the following manner. For simplicity, let us stick with the scenario where every link appears with probability $1/2$. Then the probability that, after time t, some link has not appeared is at most $m(1/2)^t$, where m is the number of edges in G. This implies that if we choose $B = O(\log(mnT))$, then with high probability, the sequence of graphs $G_1, \ldots, G_T$ is B-connected.

Thus our theorem applies to this case, albeit at the expense of some logarithmic factors due to the choice of B. We remark that it is possible to get rid of these factors by directly analyzing the decrease in $E[||z(t) - z^*||_2^2]$ coming from the random choice of graph G. Since our arguments are already quite lengthy, we do not pursue this generalization here, and refer the reader to Lobel and Ozdaglar [2010], Srivastava and Nedić [2011] where similar arguments have been made.

3.3 Numerical simulations

3.3.1 Setup

In this section, we simulate the RASGP algorithm on two classes of graphs, namely, random directed graphs and bidirectional cycle graphs. The main objective function is chosen to be a strongly convex and smooth Support Vector Machine (SVM), i.e. $F(\boldsymbol{\omega}, \gamma) = \frac{1}{2}\left(\|\boldsymbol{\omega}\|^2 + \gamma^2\right) + C_N \sum_{j=1}^{N} h(b_j(\mathbf{A}_j^\top \boldsymbol{\omega} + \gamma))$ where $\boldsymbol{\omega} \in \mathbb{R}^{d-1}$ and $\gamma \in \mathbb{R}$ are the optimization variables, and $\mathbf{A}_j \in \mathbb{R}^{d-1}, b_j \in \{-1, +1\}$, $j = 1, \ldots, N$, are the data points and their labels, respectively. The coefficient $C_N \in \mathbb{R}$ penalizes the points outside of the soft margin. We set $C_N = c/N, c = 500$ in our simulations, which depends on the total number of data points. Here, $h : \mathbb{R} \to \mathbb{R}$ is the smoothed hinge loss, initially introduced in Rennie and Srebro

[2005], defined as follows:

$$h(\xi) = \begin{cases} -0.5 - \xi, & \text{if } \xi < 0, \\ 0.5(1-\xi)^2, & \text{if } 0 \le \xi < 1, \\ 0, & \text{if } 1 \le \xi. \end{cases}$$

To solve this problem in a distributed way, we suppose all data points are spread among agents. Hence, the local objective functions are $f_i(\boldsymbol{\omega}_i, \gamma_i) = \frac{1}{2n}\left(\|\boldsymbol{\omega}\|^2 + \gamma^2\right) + C_N \sum_{j \in D_i} h(b_j(\mathbf{A}_j^\top \boldsymbol{\omega} + \gamma))$, where $D_i \subset \{1, 2, \ldots, N\}$ is an index set for data points of agent i and N is the total number of data points. We choose the size of the dataset for each local function to be a constant ($|D_i| = 50$), thus $N = 50n$. It is easy to check that each f_i has Lipschitz gradients and is strongly convex with $\mu_i = 1/n$.

We will compare our results with a centralized gradient descent algorithm, which updates every Γ_u iterations using the step-size sequence $\alpha_c(k) = \Gamma_u/(\mu k)$, in the direction of the *sum* of the gradients of all agents.

To make gradient estimates stochastic, we add a uniformly distributed noise $\boldsymbol{\varepsilon}_i \sim \mathbb{U}[-b/2, b/2]^d$ to the gradient estimates of each agent and $\boldsymbol{\varepsilon}_c \sim \mathbb{U}[-\sqrt{n}b/2, \sqrt{n}b/2]^d$ to the gradient of the centralized gradient descent, where $\mathbb{U}[b_1, b_2]^d$ denotes the uniform distribution of size d over the interval $[b_1, b_2)$, $b_1 < b_2$. Note that $\boldsymbol{\varepsilon}_i$ and $\boldsymbol{\varepsilon}_c$ are bounded and have zero mean and $\mathbb{E}[\|\boldsymbol{\varepsilon}_i\|^2] = db^2/12$ and $\mathbb{E}[\|\boldsymbol{\varepsilon}_c\|^2] = ndb^2/12$. We set $b = 4$ for all simulations.

Agents wake up with probability P_w and links fail with probability P_f, unless they reach their maximum allowed value where the algorithm forces the agent to wake up or the link to work successfully. The link delays are chosen uniformly between 1 to Γ_{del}.

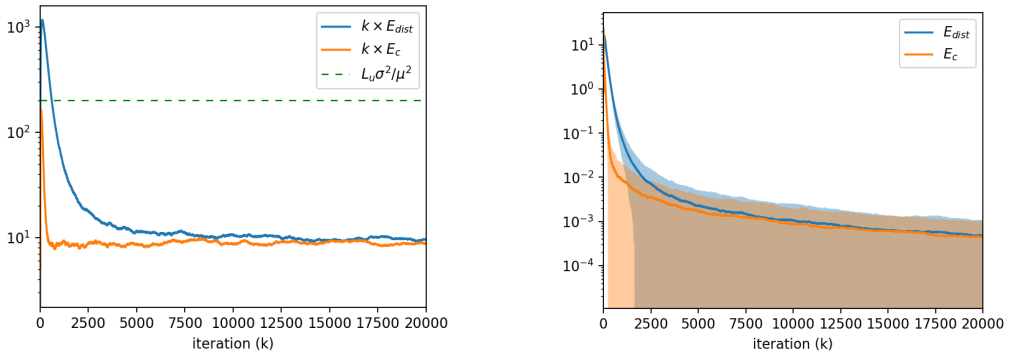
Each dataset D_i is synthetically generated by picking 25 data points around each of the centers $(1, 1)$ and $(3, 3)$ with multivariate normal distributions, labeled -1 and $+1$, respectively. In generating strongly connected random graphs, we pick each edge with a probability of 0.5 and then check if the resulting graph is strongly connected; if it isn't, we repeat the process. Since the initial step-sizes for the distributed algorithm can be very large (e.g., $\alpha(1) = 50$ for $n = 50$), to stabilize the algorithms, both algorithms are started

with $k_0 = 100$. This wouldn't affect the asymptotic convergence performance. Moreover, the initial point of the centralized algorithm and all agents in RASGP are chosen as $\mathbf{1}_d$.

Let us denote by $\hat{\mathbf{z}}(k) := (1/n)\sum_{i=1}^{n} \mathbf{z}_i(k)$ the average of $\mathbf{z}$-values of non-virtual agents. Then, we define *optimization errors* $E_{dist} := \|\hat{\mathbf{z}}(k) - \mathbf{z}^*\|^2$ and $E_c(k) := \|\mathbf{x}_c(k) - \mathbf{z}^*\|^2$ for RASGP and Centralized stochastic gradient descent, respectively.

Since our performance guarantees are for the expectation of (squared) errors, for each network setting, we perform up to 1000 Monte-Carlo simulations and use their corresponding performance to estimate the average behavior of the algorithms. Since accurately estimating the *true* expected value requires an extremely large number of simulations, in order to alleviate the effect of spikes and high variance, we take the following steps. First a batch of simulations are performed and their average is calculated. Next, to obtain a smoother plot, an average over every 100 iterations is taken. And finally, the median of these outputs over all the batches is our estimate of the expected value.

We report two figures for each setting: one including the errors E_{dist} and E_c, and another one including $k \times E_{dist}$ and $k \times E_c$ to demonstrate the convergence rates.



(a) k times squared errors. (b) Errors and 1-standard-deviation band.

Figure 3·2: Results on a directed cycle graph of size $n = 50$, synchronous with no delays and link failures ($P_w = 1$, $P_f = 0$, $\Gamma_{\text{del}} = \Gamma_f = 0, \Gamma_u = 1$, $\Gamma_s = 2$).

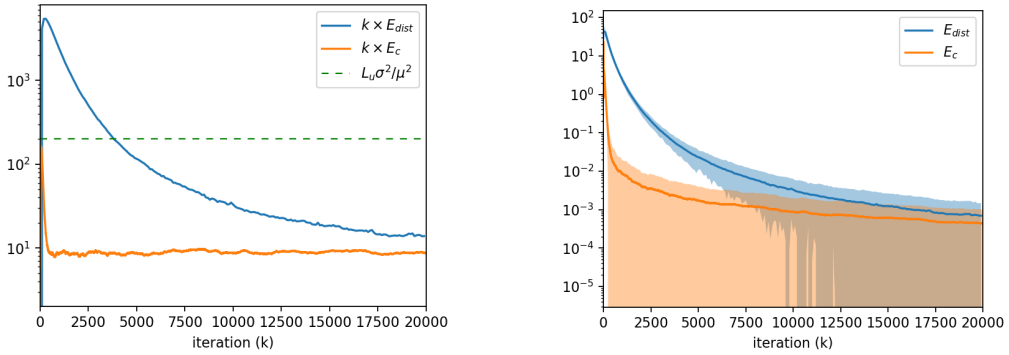
Finally, to study the non-asymptotic behavior of RASGP and its dependence on network size n, we have compared the performance of the centralized stochastic gradient descent and RASGP over a bidirectional cycle graph, with error variances of $n^2\hat{\sigma}^2$ and $\sigma_i^2 = \hat{\sigma}^2$,

respectively. Then, we plot the ratio $E_c(k)/E_{dist}(k)$ over n, for different iterations k.

3.3.2 Results

Our simulation results are consistent with our theoretical claims (due to the performance of centralized and decentralized methods growing closer over time) and show the achievement of an asymptotic network-independent convergence rate.

Fig. 3·2 shows that when there is no link failure or delay and all agents wake up at every iteration ($\Gamma_s = 2$), RASGP and centralized gradient descent have very similar performance. When we allow links to have delays and failures (see Fig. 3·3), as well as asynchronous updates (see Fig. 3·4), it takes longer for RASGP to reach its asymptotic convergence rate.

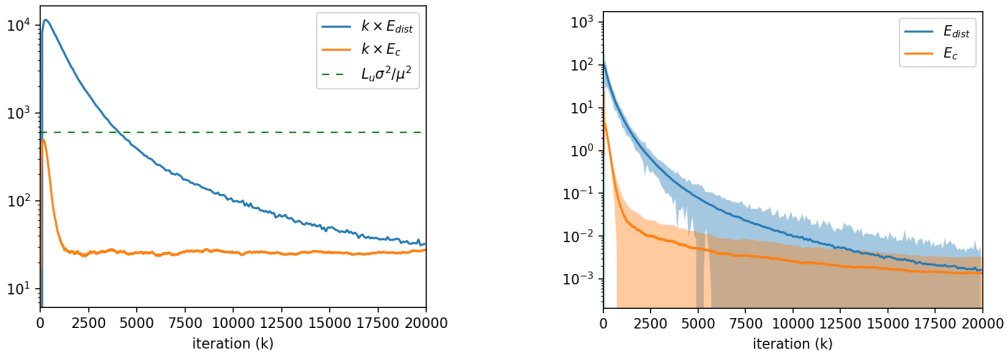


(a) k times squared errors. (b) Errors and 1-standard-deviation band.

Figure 3·3: Results on a directed cycle graph of size $n = 50$, synchronous with delays and link failures ($P_w = 1$, $P_f = 0.3$, $\Gamma_{\text{del}} = \Gamma_f = 3, \Gamma_u = 1$, $\Gamma_s = 7$).

We observe that, with all the other parameters fixed, the RASGP performs better on a random graph than on a cycle graph (see Figs. 3·4 and 3·5). A possible reason is that the cycle graph has a higher diameter or mixing time compared to the random graph, resulting in a slower decay of the consensus error.

We notice that by fixing the network size, increasing the number of iterations brings us closer to linear speed-up (see Fig. 3·6). On the other hand, when fixing the number of iterations, increasing the number of nodes, after a certain point, does not help speeding up the optimization. Moreover, by allowing link delays and failures (see Fig. 3·6(b)) we



(a) k times squared errors.

(b) Errors and 1-standard-deviation band.

Figure 3·4: Results on a directed cycle graph of size $n = 50$, asynchronous with delays and link failures ($P_w = 0.5$, $P_f = 0.3$, $\Gamma_{\text{del}} = \Gamma_f = 3, \Gamma_u = 3$, $\Gamma_s = 17$).

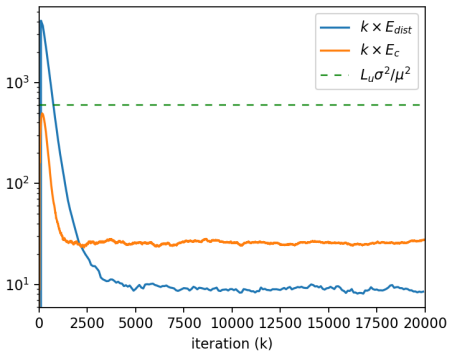
require more iterations to achieve network independence.

3.4 Conclusions

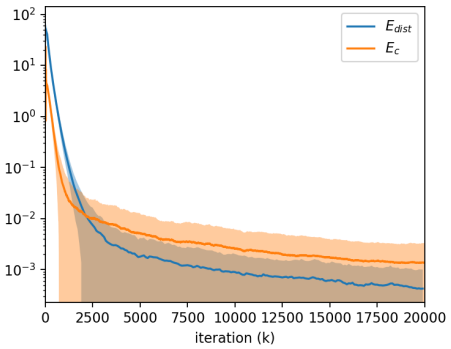
The main result of this chapter is to establish asymptotically, network independent performance for a distributed stochastic optimization method over directed graphs with message losses, delays, and asynchronous updates. Our work raises several open questions.

The most natural question raised by this chapter concerns the size of the transients. How long must the nodes wait until the network-independent performance bound is achieved? The answer, of course, will depend on the network, but also on the number of nodes, the degree of asynchrony, and the delays. Understanding how this quantity scales is required before the algorithms presented in this work can be recommended to practitioners.

More generally, it is interesting to ask which problems in distributed optimization can achieve network-independent performance, even asymptotically. For example, the usual bounds for distributed subgradient descent (see, e.g., Nedić et al. [2018]) depend on the spectral gap of the underlying network; various worst-case scalings with the number of nodes can be derived, and the final asymptotics are not network-independent. It is not immediately clear whether this is due to the analysis, or a fundamental limitation that will not be overcome.

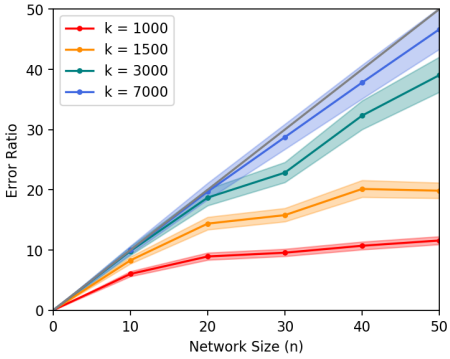


(a) k times squared errors.

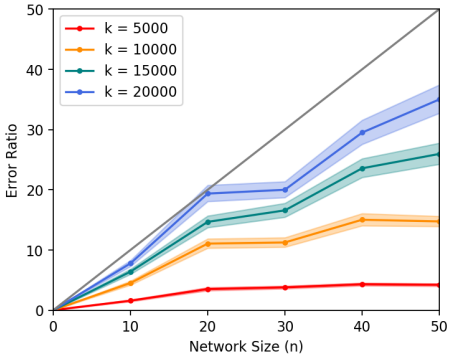


(b) Errors and 1-standard-deviation band.

Figure 3·5: Results on a directed random graph of size $n = 50$, asynchronous with delays and link failures ($P_w = 0.5$, $P_f = 0.3$, $\Gamma_{\text{del}} = \Gamma_f = 3, \Gamma_u = 3$, $\Gamma_s = 17$).



(a) Synchronous with no delays and link failures.



(b) Synchronous with delays and link failure ($P_w = 1$, $P_f = 0.3$, $\Gamma_{\text{del}} = \Gamma_f = 3, \Gamma_u = 1$, $\Gamma_s = 7$).

Figure 3·6: Error ratio over network size. Shaded areas correspond to 1-standard-deviation of the performance.

Chapter 4

Communication-efficient SGD: From Local SGD to One-Shot Averaging

4.1 Introduction

Stochastic Gradient Descent (SGD) is a widely used algorithm to minimize convex functions f in which model parameters are updated iteratively as

$$\mathbf{x}^{t+1} = \mathbf{x}^t - \eta_t \hat{\mathbf{g}}^t,$$

where $\hat{\mathbf{g}}^t$ is a stochastic gradient of f at the point $\mathbf{x}^t$ and η_t is the learning rate. This algorithm can be naively parallelized by adding more workers independently to compute a gradient and then average them at each step to reduce the variance in estimation of the true gradient $\nabla f(\mathbf{x}^t)$ [Dekel et al., 2012]. This method requires each worker to share their computed gradients with each other at every iteration. We will refer to this method as "synchronized parallel SGD."

However, it is widely acknowledged that communication is a major bottleneck of this method for large scale optimization applications [McMahan et al., 2017, Konečnỳ et al., 2016, Lin et al., 2018b]. Often, mini-batch parallel SGD is suggested to address this issue by increasing the computation to communication ratio. Nonetheless, too large mini-batch size might degrade performance [Lin et al., 2018a]. Along the same lines of increasing the computation over communication effort, *local* SGD has been proposed to reduce communications [McMahan et al., 2017, Dieuleveut and Patel, 2019]. In this method, workers compute (stochastic) gradients and update their parameters locally, and communicate only once in a while to obtain the average of their parameters. Local SGD improves the com-

munication efficiency not only by reducing the number of communication rounds, but also alleviates the synchronization delay caused by waiting for slow workers and evens out the variations in workers' computing time [Wang and Joshi, 2018b].

On the other hand, since individual gradients of each worker are calculated at different points, this method introduces residual error as opposed to fully synchronized SGD. Therefore, there is a trade-off between having fewer communication rounds and introducing additional errors to the gradient estimates.

The idea of making local updates is not new and has been used in practice for a while [Konečnỳ et al., 2016]. However, until recently, there have been few successful efforts to analyze Local SGD theoretically and therefore it is not fully understood yet. Zhang et al. [2016] show that for quadratic functions, when the variance of the noise is higher far from the optimum, frequent averaging leads to faster convergence. The first question we try to answer in this work is: how many communication rounds are needed for Local SGD to have the *similar* convergence rate of a synchronized parallel SGD while achieving performance that linearly improves in the number of workers?

Stich [2019] was among the first who sought to answer this question for general strongly convex and smooth functions and showed that the communication rounds can be reduced up to a factor of $H = \mathcal{O}(\sqrt{T/N})$, without affecting the asymptotic convergence rate (up to constant factors), where T is the total number of iterations and N is number of parallel workers.

Focusing on smooth and possibly non-convex functions which satisfy a Polyak-Lojasiewicz condition, Haddadpour et al. [2019] demonstrate that only $R = \Omega((TN)^{1/3})$ communication rounds are sufficient to achieve asymptotic performance that scales proportionately to $1/N$.

More recently, Khaled et al. [2020] and Stich and Karimireddy [2019] improve upon the previous works by showing linear speed-up for Local SGD with only $\Omega\left(N \text{ poly log } (T)\right)$ communication rounds when data is identically distributed among workers and f is strongly convex. Their works also consider the cases when f is not necessarily strongly convex as well as the case of data being heterogeneously distributed among workers.

One-Shot Averaging (OSA), a method that takes an extreme approach to reducing communication, involves workers performing local updates until the very end when they average their parameters [Mcdonald et al., 2009, Zinkevich et al., 2010, Zhang et al., 2013c, Rosenblatt and Nadler, 2016, Godichon-Baggioni and Saadane, 2020]. This method can be seen as an extreme case of Local SGD with $R = 1$ and $H = T$ local steps. Dieuleveut and Patel [2019], Godichon-Baggioni and Saadane [2020] provide an analysis of OSA and show that asymptotically, linear speed-up in the number of workers is achieved for a weighted average of iterates. However, both of these works make restrictive assumptions such as uniformly three-times continuously differentiability and bounded second and third derivatives or twice differentiability almost everywhere with bounded Hessian, respectively. The second question we attempt to answer in this work, is whether these assumptions can be relaxed and OSA can achieve linear speed-up in more general scenarios.

In this work, we focus on smooth and strongly convex functions with a general noise model. Our contributions are three-fold:

1. We propose a communication strategy which requires only $R = \Omega(N)$ communication rounds to achieve performance that scales as $1/N$ in the number of workers. To the best of the authors' knowledge, this is the only work to show that the number of communications can be taken to be completely independent of T. All previous works required a number of communications which was at least N times a polynomial in $\log(T)$, or had a stronger scaling with T. A comparison of our result to the available literature can be found in Table 4.1.

2. We show under mild additional assumptions, in particular twice differentiability on a neighborhood of the optimal point, OSA reaches linear speed-up asymptotically, i.e., with only one communication round we achieve the convergence rate of $\mathcal{O}(1/(NT))$.

3. We simulate a simple example which is not twice differentiable at the optimum and observe that our bounds for part 1. are reasonably close to being tight. In particular, using 1 or $\sqrt{N}$ or $N^{3/4}$ communications does not appear to result in a linear speed-up

Table 4.1: Comparison of Similar Works

Reference	Convergence rate $f(\hat{\mathbf{x}}^T) - f^*$[a]	Communication Rounds R	Noise model
Stich [2019]	$\mathcal{O}(\frac{\xi^0}{R^3} + \frac{\sigma^2}{\mu NT} + \frac{\kappa G^2}{\mu R^2})$[b]	$\Omega(\sqrt{TN})$	uniform
Haddadpour et al. [2019]	$\mathcal{O}(\frac{\xi^0}{R^3} + \frac{\kappa\sigma^2}{\mu NT} + \frac{\kappa^2\sigma^2}{\mu NTR})$	$\Omega((TN)^{1/3})$	uniform with strong-growth[c]
Stich and Karimireddy [2019]	$\tilde{\mathcal{O}}(\frac{\kappa NH\xi^0}{\exp(R/(\kappa N))} + \frac{\sigma^2}{\mu NT})$[d]	$\Omega(N * \text{poly-log}(T))$	uniform with strong-growth
Khaled et al. [2020]	$\tilde{\mathcal{O}}(\frac{\kappa\xi^0}{T^2} + \frac{\kappa\sigma^2}{\mu NT} + \frac{\kappa^2\sigma^2}{\mu TR})$	$\Omega(N * \text{poly-log}(T))$	uniform
This work	$\mathcal{O}(\frac{(1+c\kappa^2\ln(TR^{-2}))\xi^0}{\kappa^{-2}T^2} + \frac{\kappa\sigma^2}{\mu NT} + \frac{\kappa^2\sigma^2}{\mu TR})$[e]	$\Omega(N)$	uniform with strong-growth

[a] Depending on the work, $\hat{\mathbf{x}}^T$ is either the last iterate or a weighted average of iterates up to T.
[b] G is the uniform upper bound assumed for the l_2 norm of gradients in the corresponding work.
[c] This noise model is defined in Assumption 8.
[d] $\tilde{\mathcal{O}}(.)$ ignores the poly-logarithmic and constant factors.
[e] c is the multiplicative factor in the noise model defined in Assumption 8.

in the number of workers (while N communications does give a linear speed-up).

The rest of this chapter is organized as follows. In the following subsection we outline the related literature and ongoing works. In Section 4.2 we define the main problem and state our assumptions. We present our theoretical findings in Sections 4.3 and 4.4 followed by numerical experiments in Section 4.5 and conclusion remarks in Section 4.6.

4.1.1 Related works

There has been a lot of effort in the recent research to take into account the communication delays and training time in designing faster algorithms [McDonald et al., 2010, Zhang et al., 2015, Bijral et al., 2016, Kairouz et al., 2019]. See Tang et al. [2020] for a comprehensive survey of communication efficient distributed training algorithms considering both system-level and algorithm-level optimizations.

Many works study the communication complexity of distributed methods for convex optimization [Arjevani and Shamir, 2015, Woodworth et al., 2020b] and statistical estimation [Zhang et al., 2013b]. Woodworth et al. [2020b] present a rigorous comparison of Local SGD with H local steps and mini-batch SGD with H times larger mini-batch size and the same number of communication rounds (we will refer to such a method as large mini-batch

SGD) and show regimes in which each algorithm performs better: they show that Local SGD is strictly better than large mini-batch SGD when the functions are quadratic. Moreover, they prove a lower bound on the worst case of Local SGD that is higher than the worst-case error of large mini-batch SGD in a certain regime. Zhang et al. [2013b] study the minimum amount of communication required to achieve centralized minimax-optimal rates by establishing lower bounds on minimax risks for distributed statistical estimation under a communication budget.

A parallel line of work studies the convergence of Local SGD with non-convex functions [Zhou and Cong, 2018]. Yu et al. [2019] was among the first works to present provable guarantees of Local SGD with linear speed-up. Wang and Joshi [2018b] and Koloskova et al. [2020] present unified frameworks for analyzing decentralized SGD with local updates, elastic averaging or changing topology. The follow-up work of Wang and Joshi [2018a] presents ADACOMM, an adaptive communication strategy that starts with infrequent averaging and then increases the communication frequency in order to achieve a low error floor. They analyze the error-runtime trade-off of Local SGD with nonconvex functions and propose communication times to achieve faster runtime.

Another line of work reduces the communication by compressing the gradients and hence limiting the number of bits transmitted in every message between workers [Lin et al., 2018b, Alistarh et al., 2017, Wangni et al., 2018, Stich et al., 2018, Stich and Karimireddy, 2019].

Asynchronous methods have been studied widely due to their advantages over synchronized methods which suffer from synchronization delays due to the slower workers [Spiridonoff et al., 2020]. Wang et al. [2019] study the error-runtime trade-off in decentralized optimization and proposes MATCHA, an algorithm which parallelizes inter-node communication by decomposing the topology into matchings. However, these methods are relatively more involved and they often require full knowledge of the network, solving a semi-definite program and/or calculating communication probabilities (schedules) as in Hendrikx et al. [2019].

The homogeneous data assumption. In this work, we focus on the case when the data distribution is the same across workers. A number of previous works [Khaled et al., 2020, Haddadpour et al., 2019, Stich, 2019, Dieuleveut and Patel, 2019] studied local SGD under this assumption. Sharing the data set across multiple workers in this way is a popular strategy to speed up training. For example, such data sharing is implemented in Chen et al. [2012], Yadan et al. [2013], Zhang et al. [2013a] to speed up training of deep neural networks with multiple GPUs within a single sever. While there are many widely used mechanisms such as Horovod [Sergeev and Del Balso, 2018] for *synchronous* data-parallel distributed training, they share a major communication bottleneck of broadcasting gradients to all workers [Grubic et al., 2018]. Local SGD improves on these methods by reducing the communication of model parameters from every iteration to a smaller number of rounds during the entire optimization process. Our approach further reduces the communication overhead by communicating less as the number of iterations grows.

4.2 Problem formulation

Suppose there are N workers $\mathcal{V} = \{1, \ldots, N\}$, trying to minimize $f : \mathbb{R}^d \to \mathbb{R}$ in parallel. We assume all workers have access to f through noisy gradients. In Local SGD, workers perform local gradient steps and occasionally calculate the average of all workers' iterates. Each worker i holds a local parameter $\mathbf{x}_i^t$ at iteration t. There is a set $\mathcal{I} \subset [T]$ of communication times and nodes perform the following update:

$$\mathbf{x}_i^{t+1} = \begin{cases} x_i^t - \eta_t \hat{\mathbf{g}}_i^t, & \text{if } t+1 \notin \mathcal{I}, \\ \frac{1}{N}\sum_{j=1}^N (\mathbf{x}_j^t - \eta_t \hat{\mathbf{g}}_j^t), & \text{if } t+1 \in \mathcal{I}, \end{cases} \tag{4.1}$$

where $\hat{\mathbf{g}}_i^t$ is an unbiased stochastic gradient of f at $\mathbf{x}_i^t$. When $\mathcal{I} = [T]$, we recover fully synchronized parallel SGD while $\mathcal{I} = \{T\}$ recovers one-shot averaging. Let us denote by $\bar{\mathbf{x}}^t := (\sum_{i=1}^N \mathbf{x}_i^t)/N$ the average of the iterates of all workers. Notice that $\mathbf{x}_i^t = \bar{\mathbf{x}}^t$ for $t \in \mathcal{I}$ and $i \in \mathcal{V}$. Pseudo-code for Local SGD is provided as Algorithm 4.

Next we state the assumptions that we will use in our results. Note that we will not

Algorithm 4 Local SGD

Input: $\mathbf{x}_i^0 = \mathbf{x}^0$ for all $i \in [n]$, total number of iterations T, the step-size sequence $\{\eta_t\}_{t=0}^{T-1}$, and $\mathcal{I} \subseteq [T]$
for $t = 0, \ldots, T-1$ **do**
  for $j = 1, \ldots, N$ **do**
    evaluate a stochastic gradient $\hat{\mathbf{g}}_j^t$
    if $t+1 \in \mathcal{I}$ **then**
      $\mathbf{x}_j^{t+1} = \frac{1}{N}\sum_{i=1}^N (\mathbf{x}_i^t - \eta_t \hat{\mathbf{g}}_i^t)$
    else
      $\mathbf{x}_j^{t+1} = \mathbf{x}_j^t - \eta_t \hat{\mathbf{g}}_j^t$
    end if
  end for
end for

require all of them to hold at once.

Assumption 4 (smoothness)**.** *The function $f : \mathbb{R}^d \to \mathbb{R}$ is continuously differentiable and its gradients are L-Lipschitz, i.e.,*

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \leq L\|\mathbf{x} - \mathbf{y}\|, \qquad \forall \mathbf{x}, \mathbf{y}.$$

Assumption 5 (strong convexity)**.** *f is μ-strongly convex with $\mu > 0$, i.e.,*

$$f(\mathbf{x}) + \langle \mathbf{g}, \mathbf{y} - \mathbf{x} \rangle + \frac{\mu}{2}\|\mathbf{x} - \mathbf{y}\|^2 \leq f(\mathbf{y}), \qquad \forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^d, \forall \mathbf{g} \in \partial f(\mathbf{x}),$$

where $\partial f(\mathbf{x})$ denotes the set of subgradients of f at $\mathbf{x}$. When f is also continuously differentiable, $\partial f(\mathbf{x}) = \{\nabla f(\mathbf{x})\}$.

Note that when f satisfies Assumption 5, it has a *unique* optimal point $\mathbf{x}^*$ where $f(\mathbf{x}^*) = f^*$ where $f^* = \min_{\mathbf{x}} f(\mathbf{x})$.

Assumption 6 (Polyak-Łohasiewicz condition)**.** *f is μ-Polyak-Łohasiewicz (μ-PL for short) if*

$$\|\nabla f(\mathbf{x})\|^2 \geq 2\mu(f(\mathbf{x}) - f^*), \qquad \forall \mathbf{x}.$$

where $f^* = \min_{\mathbf{x}} f(\mathbf{x})$ is the global minimum of f. We further assume that f has a unique *optimal point $\mathbf{x}^*$ where $f(\mathbf{x}^*) = f^*$.*

When f satisfies both Assumptions 4 and 5 or Assumptions 4 and 6, we define $\kappa = L/\mu$ as the condition number of f.

Strong convexity implies the PL condition but the reverse does not always hold. For instance, the logistic regression loss function satisfies the PL condition over any compact set [Karimi et al., 2016]. In fact, a PL function is not even necessarily convex. Charles and Papailiopoulos [2018] show that deep networks with linear activation functions are PL almost everywhere in the parameter space. Allen-Zhu et al. [2018] show, with high probability over random initializations, that sufficiently wide recurrent neural networks satisfy the PL condition. Therefore, the PL condition is more applicable, especially in the context of neural networks [Madden et al., 2020].

Assumption 7 (twice differentiability at the optimum)**.** *f is twice continuously differentiable on an open set containing the optimal point $\mathbf{x}^*$.*

We make the following assumption on the noise of stochastic gradients, using $\mathbf{w}_i^t = \hat{\mathbf{g}}_i^t - \nabla f(\mathbf{x}_i^t)$ to denote the difference between the stochastic and true gradients.

Assumption 8 (uniform with strong-growth noise)**.** *Conditioned on the iterate $\mathbf{x}_i^t$, the random variable $\mathbf{w}_i^t$ is zero-mean and independent with its expected squared norm error bounded as,*

$$\mathbb{E}[\|\mathbf{w}_i^t\|^2 | \mathbf{x}_i^t] \leq c\|\nabla f(\mathbf{x}_i^t)\|^2 + \sigma^2,$$

where $\sigma^2, c \geq 0$ are constants.

The noise model of Assumption 8 is very general and it includes the common case with uniformly bounded squared norm error when $c = 0$. As it is noted by Zhang et al. [2016], the advantage of periodic averaging compared to one-shot averaging only appears when c/σ^2 is large. Therefore, to study Local SGD, it is important to consider a noise model as in Assumption 8 to capture the effects of frequent averaging. Among the related works mentioned in Table 4.1, only Stich and Karimireddy [2019] and Haddadpour et al. [2019] analyze this noise model while the rest study the special case with $c = 0$. SGD under this noise model with $c > 0$ and $\sigma^2 = 0$ was first studied in Schmidt and Roux [2013] under the name *strong-growth condition*. Therefore we refer to the noise model considered in this work as *uniform with strong-growth*.

Assumption 9 (sub-Gaussian noise)**.** *Conditioned on the iterate* $\mathbf{x}_i^t$*, random variable* $\mathbf{w}_i^t$ *is zero-mean, independent and* $[\mathbf{w}_i^t]_l$ *is* $(\sigma/\sqrt{d})$*-sub-Gaussian, for* $l = 1, \ldots, d$*, i.e.,*

$$\mathbb{E}[\exp(\lambda([\mathbf{w}_i^t]_l - \mathbb{E}[\mathbf{w}_i^t]_l))|\mathbf{x}_i^t] \leq \exp\left(\frac{\lambda^2\sigma^2}{2d}\right), \qquad \forall \lambda \in \mathbb{R}, l = 1, \ldots, d.$$

Thus, it has uniformly bounded variance $\mathbb{E}[\|\mathbf{w}_i^t\|^2|\mathbf{x}_i^t] \leq \sigma^2$*.*

A sub-Gaussian noise model is commonly assumed for deriving concentration bounds for SGD, which we will use to prove our results for OSA.

As already mentioned in the Introduction, the main goal of this chapter is to study the effect of communication times on the convergence of the Local SGD and provide better theoretical guarantees. In what follows, we claim that by carefully choosing the communication times, linear speed-up of parallel SGD can be attained with only a small number of communication instances. Moreover, we will obtain a set of sufficient conditions for OSA to achieve linear speed-up.

4.3 Convergence results for Local SGD

In this section we present our main convergence results for Local SGD. Before stating our main result in Theorem 6, we present the following theorem which will be used later to prove our main result as well as to better understand the choice of varying number of local steps. In the following theorem, we show an upper bound for the sub-optimality error, in the sense of function value, for any choice of communication times $\mathcal{I}$. Theorem 6 will be obtained by specializing the following bound. First, let us introduce the notation

$$0 = \tau_0 < \tau_1 < \ldots < \tau_R = T,$$

for the communication times. Further, let us define $H_i := \tau_{i+1} - \tau_i$ to be the i'th intercommunication interval and denote the most recent communication time by $\tau(t) := \max\{t' \in \mathcal{I} | t' \leq t\}$.

Theorem 5. *Suppose Assumptions 4, 5 and 8 hold. Choose $\beta \geq 9\kappa$ and communication times $\mathcal{I} = \{\tau_i | i = 1, \ldots, R\}$ such that it holds for $i = 0, \ldots, R-1$,*

$$12\kappa^2 c \ln(1 + \frac{H_i - 1}{\tau_i + \beta}) + 3\kappa(1 + \frac{c}{N}) - (\tau_i + \beta) \leq 0. \tag{4.2}$$

Set step-sizes $\eta_t = 3/(\mu(t+\beta))$, $t = 0, 1, \ldots, T-1$. Then, using Algorithm 4, we have

$$\mathbb{E}[f(\bar{\mathbf{x}}^T)] - f^* \leq \frac{\beta^2(f(\bar{\mathbf{x}}^0) - f^*)}{T^2} + \frac{9L\sigma^2}{2\mu^2 NT} + \frac{18L^2\sigma^2}{\mu^3 T^2} \sum_{t=0}^{T-1} \frac{t - \tau(t)}{t + \beta}, \tag{4.3}$$

The last term in Equation (4.3) is due the to disagreement between workers (consensus error), introduced by local computations without any communication. As the inter-communication intervals become larger, $t - \tau(t)$ becomes larger as well and increases the overall optimization error. This term explains the trade-off between communication efficiency and the optimization error.

Note that condition (4.2) is mild. For instance, it suffices to set $\beta \geq \max\{12\kappa^2 c \ln(1 + T/(9\kappa)) + 3\kappa(1 + c/N), 9\kappa\}$. Moreover, the bound in (4.3) is for the last iterate T, and does not require keeping track of a weighted average of all the iterates.

Theorem 5 not only bounds the optimization error, but introduces a methodological approach to select the communication times to achieve smaller errors. For the scenarios when the user can afford to have a certain number of a communications, they can select τ_i to minimize the last term in (4.3).

One-shot averaging. Plugging $H = T$ in Theorem 5, we obtain a convergence rate of $\mathcal{O}(\kappa^2\sigma^2/(\mu T))$ without any linear speed-up. Among previous works, only Khaled et al. [2020] shows a similar result.

4.3.1 Fixed-length intervals

A simple way to select the communication times $\mathcal{I}$, is to split the whole training time T to R intervals of length at most H. Then we can use the following bound in Equation (4.3),

$$\sum_{t=0}^{T-1} \frac{t-\tau(t)}{t+\beta} \leq (H-1) \sum_{t=0}^{T-1} \frac{1}{t+\beta} \leq (H-1)\ln(1+\frac{T}{\beta-1}).$$

We state this result formally in the following corollary.

Corrollary 5. *Suppose assumptions of Theorem 5 hold and in addition, workers communicate at least once every H iterations. Then,*

$$\mathbb{E}[f(\bar{\mathbf{x}}^T)] - f^* \leq \frac{\beta^2(f(\bar{\mathbf{x}}^0)-F^*)}{T^2} + \frac{9L\sigma^2}{2\mu^2 NT} + \frac{18L^2\sigma^2(H-1)}{\mu^3 T^2}\ln(1+\frac{T}{\beta-1}). \quad (4.4)$$

Linear speed-up. Setting $H = \mathcal{O}(T/(N\ln(T)))$ we achieve linear speed-up in the number of workers, which is equivalent to a communication complexity of $R = \Omega(N\ln(T))$. To the best of the authors' knowledge, this is the tightest communication complexity that is shown to achieve linear speed-up. Khaled et al. [2020] and Stich and Karimireddy [2019] have shown a similar communication complexity.

Recovering synchronized SGD. When $H = 1$, the last term in (4.4) disappears and we recover the convergence rate of parallel SGD, albeit, with a worse dependence on κ.

4.3.2 Linearly growing intervals

Here, we present our main result for Local SGD.

Theorem 6. *Suppose Assumptions 4 (smoothness), 5 (strong convexity) and 8 (uniform with strong growth noise) hold.*
Choose the parameters as follows: R such that $1 \leq R \leq \sqrt{2T}$ and $a := \lceil 2T/R^2 \rceil \geq 1$, $H_i = a(i+1)$ and $\tau_{i+1} = \min(\tau_i + H_i, T)$ for $i = 0, \ldots, R-1$. Choose

$$\beta \geq \max\{9\kappa, 12\kappa^2 c \max\{\ln(3), \ln(1 + T/(4\kappa R^2))\} + 3\kappa(1 + c/N)\}$$

and set the learning rate as $\eta_t = 3/\mu(t+\beta), t = 0, 1, \ldots, T-1$. Then using Algorithm 4 we have,

$$\mathbb{E}[f(\bar{\mathbf{x}}^T)] - f^* \leq \frac{\beta^2(f(\bar{\mathbf{x}}^0) - f^*)}{T^2} + \frac{9L\sigma^2}{2\mu^2 NT} + \frac{144L^2\sigma^2}{\mu^3 RT}.$$

Corrollary 6. *Under the assumptions of Theorem 6, selecting the number of communications $R = \Omega(\kappa N)$ we obtain*

$$\mathbb{E}[f(\bar{\mathbf{x}}^T)] - f^* \leq \frac{\beta^2(f(\bar{\mathbf{x}}^0) - f^*)}{T^2} + \mathcal{O}\left(\frac{L\sigma^2}{\mu^2 NT}\right).$$

The choice of communication times in Theorem 6 aligns with the intuition that workers need to communicate more frequently at the beginning of the optimization. As the the step-sizes become smaller and workers' local parameters get closer to the global minimum, they diverge more slowly from each other and therefore, less communication is required to re-align them. The advantage of this communication strategy over fixed periodic averaging has been only empirically shown in Haddadpour et al. [2019].

4.3.3 Proof of results for Local SGD

Here we give an outline of the proofs for the Local SGD results presented in this chapter. Let us define the following notations used in the proofs presented here.

$$\bar{\mathbf{g}}^t := \frac{1}{N}\sum_{i=1}^{n} \mathbf{g}_i^t, \qquad G^t := \frac{1}{N}\sum_{i=1}^{N} \|\mathbf{g}_i^t\|^2, \qquad \mathbf{w}_i^t := \hat{\mathbf{g}}_i^t - \mathbf{g}_i^t.$$

Moreover, define $\mathcal{F}^t := \{\mathbf{x}_i^k, \hat{\mathbf{g}}_i^k | 1 \leq i \leq N, 0 \leq k \leq t-1\} \cup \{\mathbf{x}_i^t | 1 \leq i \leq N\}$.

Perturbed iterates. A common approach in analyzing parallel algorithms such as Local SGD is to study the evolution of the sequence $\{\bar{\mathbf{x}}^t\}_{t\geq 0}$. We have,

$$\bar{\mathbf{x}}^{t+1} = \bar{\mathbf{x}}^t - \frac{\eta_t}{N}\sum_{i=1}^{N} \hat{\mathbf{g}}_i^t = \bar{\mathbf{x}}^t - \eta_t \tilde{\mathbf{g}}^t, \tag{4.5}$$

where $\tilde{\mathbf{g}}^t := (\sum_{i=1}^{N} \hat{\mathbf{g}}_i^t)/N$ is the average of the stochastic gradient estimates of all workers.

Let us define $\xi^t := \mathbb{E}[f(\bar{\mathbf{x}}^t)] - f^*$ to be the optimality error. The following lemma, which is similar to a part of the proof found in Haddadpour et al. [2019], bounds the optimality error at each iteration recursively.

Lemma 31. *Let Assumptions 4, 5 and 8 hold. Then,*

$$\xi^{t+1} \leq \xi^t(1-\mu\eta_t) + \frac{L^2\eta_t}{2N}\mathbb{E}\left[\sum_{i=1}^{N} \|\bar{\mathbf{x}}^t - \mathbf{x}_i^t\|^2\right] + \frac{\eta_t^2 L}{2}\mathbb{E}[\|\tilde{\mathbf{g}}^t\|_2^2] - \frac{\eta_t}{2N}\mathbb{E}\left[\sum_{i=1}^{N} \|\nabla f(\mathbf{x}_i^t)\|^2\right].$$

Proof. By Assumptions 4 and 5 and (4.5) we have,

$$\mathbb{E}[f(\bar{\mathbf{x}}^{t+1}) - f(\bar{\mathbf{x}}^t)] \leq -\eta_t \mathbb{E}[\langle \nabla f(\bar{\mathbf{x}}^t), \tilde{\mathbf{g}}^t\rangle] + \frac{\eta_t^2 L}{2}\mathbb{E}[\|\tilde{\mathbf{g}}^t\|_2^2]. \tag{4.6}$$

We bound the first term on the R.H.S of (4.6) by conditioning on $\mathcal{F}^t$ as follows:

$$\begin{aligned}\mathbb{E}[\langle \nabla f(\bar{\mathbf{x}}^t), \tilde{\mathbf{g}}^t \rangle | \mathcal{F}^t] &= \frac{1}{N}\sum_{i=1}^{N} \langle \nabla f(\bar{\mathbf{x}}^t), \mathbb{E}[\hat{\mathbf{g}}_i^t | \mathbf{x}_i^t] \rangle \\ &= \frac{1}{2}\|\nabla f(\bar{\mathbf{x}}^t)\|^2 + \frac{1}{2N}\sum_{i=1}^{N}\|\nabla f(\mathbf{x}_i^t)\|^2 - \frac{1}{2N}\sum_{i=1}^{N}\|\nabla f(\bar{\mathbf{x}}^t) - \nabla f(\mathbf{x}_i^t)\|^2 \\ &\geq \mu(f(\bar{\mathbf{x}}^t) - f^*) + \frac{1}{2N}\sum_{i=1}^{N}\|\nabla f(\mathbf{x}_i^t)\|^2 - \frac{L^2}{2N}\sum_{i=1}^{N}\|\bar{\mathbf{x}}^t - \mathbf{x}_i^t\|^2, \end{aligned} \tag{4.7}$$

where we used $\langle a, b\rangle = \frac{1}{2}\|a\|^2 + \frac{1}{2}\|b\|^2 - \frac{1}{2}\|a-b\|^2$ in the second equation and $(1/2)\|\nabla f(\mathbf{x})\|^2 \geq \mu(f(\mathbf{x}) - f^*)$ as well as smoothness of f in the last inequality. Taking full expectation of (4.7) and combining it with (4.6) concludes the lemma. □

Equipped with Lemma 31, we can bound the consensus error ($\mathbb{E}[\sum_{i=1}^{N}\|\bar{\mathbf{x}}^t - \mathbf{x}_i^t\|^2]$) as well as the term $\mathbb{E}[\|\tilde{\mathbf{g}}^t\|^2]$ in the following lemmas.

Consensus error. In the following lemmas, we utilize the structure of the problem to bound the consensus error recursively. Let us define $\mathbf{g}_i^t = \nabla f(\mathbf{x}_i^t)$ as the true gradient at worker i's iterate at time t.

Lemma 32. *Let Assumptions 4, 5 and 8 hold. Then,*

$$\begin{aligned}\mathbb{E}\left[\sum_{i=1}^{N}\|\mathbf{x}_i^{t+1} - \bar{\mathbf{x}}^{t+1}\|^2\right] \leq \mathbb{E}\left[\sum_{i=1}^{N}\|\mathbf{x}_i^t - \bar{\mathbf{x}}^t\|^2\right](1 - \eta_t\mu + \eta_t^2\mu L) \\ + (N-1)\eta_t^2\sigma^2 + \left(1 - \frac{1}{N}\right)\eta_t^2 c\mathbb{E}\left[\sum_{i=1}^{N}\|\mathbf{g}_i^t\|^2\right]. \end{aligned} \tag{4.8}$$

Before proving this lemma, we state an important identity in the following lemma.

Lemma 33. *Let $\mathbf{u}_1, \dots \mathbf{u}_n \in \mathbb{R}^d$ be n arbitrary vectors. Define $\bar{\mathbf{u}} = (\sum_{i=1}^{n}\mathbf{u}_i)/n$. Then,*

$$\sum_{i=1}^{n}\|\mathbf{u}_i - \bar{\mathbf{u}}\|^2 = \sum_{i=1}^{n}\|\mathbf{u}_i\|^2 - n\|\bar{\mathbf{u}}\|^2.$$

Proof. We have

$$\sum_{i=1}^{n}\|\mathbf{u}_i-\bar{\mathbf{u}}\|^2=\sum_{i=1}^{n}\|\mathbf{u}_i\|^2+n\|\bar{\mathbf{u}}\|^2-2\sum_{i=1}^{n}\langle\mathbf{u}_i,\bar{\mathbf{u}}\rangle$$
$$=\sum_{i=1}^{n}\|\mathbf{u}_i\|^2+n\|\bar{\mathbf{u}}\|^2-2n\langle\bar{\mathbf{u}},\bar{\mathbf{u}}\rangle=\sum_{i=1}^{n}\|\mathbf{u}_i\|^2-n\|\bar{\mathbf{u}}\|^2.$$

□

Proof of Lemma 32. We have,

$$\sum_{i=1}^{N}\mathbb{E}\left[\left\|\mathbf{x}_i^{t+1}-\bar{\mathbf{x}}^{t+1}\right\|^2\right]=\sum_{i=1}^{N}\left\|\mathbb{E}\left[\mathbf{x}_i^{t+1}-\bar{\mathbf{x}}^{t+1}\right]\right\|^2$$
$$+\sum_{i=1}^{N}\mathbb{E}\left[\left\|\mathbf{x}_i^{t+1}-\bar{\mathbf{x}}^{t+1}-\mathbb{E}\left[\mathbf{x}_i^{t+1}-\bar{\mathbf{x}}^{t+1}\right]\right\|^2\right]. \quad (4.9)$$

Let us consider the first term on the right hand side of (4.9). Taking conditional expectation of both sides of (4.5) implies,

$$\sum_{i=1}^{N}\left\|\mathbb{E}\left[\mathbf{x}_i^{t+1}-\bar{\mathbf{x}}^{t+1}\middle|\,\mathcal{F}^t\right]\right\|^2=\sum_{i=1}^{N}\|\mathbf{x}_i^t-\bar{\mathbf{x}}^t-\eta_t(\mathbf{g}_i^t-\bar{\mathbf{g}}^t)\|^2$$
$$=\sum_{i=1}^{N}\left(\|\mathbf{x}_i^t-\bar{\mathbf{x}}^t\|^2+\eta_t^2\|\mathbf{g}_i^t-\bar{\mathbf{g}}^t\|^2-2\eta_t\langle\mathbf{g}_i^t,\mathbf{x}_i^t-\bar{\mathbf{x}}^t\rangle\right). \quad (4.10)$$

By L-smoothness of F, $\|\nabla f(\mathbf{x})-\nabla f(\mathbf{y})\|^2\leq 2L(f(\mathbf{x})-f(\mathbf{y})-\langle\nabla f(\mathbf{y}),\mathbf{x}-\mathbf{y}\rangle)$. Thus,

$$\sum_{i=1}^{N}\|\mathbf{g}_i^t-\bar{\mathbf{g}}^t\|^2\leq\sum_{i=1}^{N}\|\mathbf{g}_i^t-\nabla f(\bar{\mathbf{x}}^t)\|^2\leq$$
$$\sum_{i=1}^{N}2L\left(f(\bar{\mathbf{x}}^t)-f(\mathbf{x}_i^t)-\langle\mathbf{g}_i^t,\bar{\mathbf{x}}^t-\mathbf{x}_i^t\rangle\right)\leq 2L\sum_{i=1}^{N}\langle\mathbf{g}_i^t,\mathbf{x}_i^t-\bar{\mathbf{x}}^t\rangle. \quad (4.11)$$

Moreover, by μ-strong convexity of F,

$$\sum_{i=1}^{N}\langle\mathbf{g}_i^t,\mathbf{x}_i^t-\bar{\mathbf{x}}^t\rangle\geq\sum_{i=1}^{N}\left(f(\mathbf{x}_i^t)-f(\bar{\mathbf{x}}_i^t)+\frac{\mu}{2}\|\mathbf{x}_i^t-\bar{\mathbf{x}}^t\|^2\right)\geq\frac{\mu}{2}\sum_{i=1}^{N}\|\mathbf{x}_i^t-\bar{\mathbf{x}}^t\|^2. \quad (4.12)$$

We used the Jensen's inequality $\sum_{i=1}^{N}f(\mathbf{x}_i^t)-f(\bar{\mathbf{x}}^t)\leq 0$ in both equations above. Combining

(4.10)-(4.12) and having $\eta_t < 1/L$ we obtain,

$$\sum_{i=1}^{N} \|\mathbb{E}[\mathbf{x}_i^{t+1} - \bar{\mathbf{x}}^{t+1}|\mathcal{F}^t]\|^2 \leq \sum_{i=1}^{N} \|\mathbf{x}_i^t - \bar{\mathbf{x}}^t\|^2 - (2\eta_t - 2\eta_t^2 L) \sum_{i=1}^{N} \langle \mathbf{g}_i^t, \mathbf{x}_i^t - \bar{\mathbf{x}}^t \rangle$$
$$\leq \sum_{i=1}^{N} \|\mathbf{x}_i^t - \bar{\mathbf{x}}^t\|^2 \left(1 - \eta_t \mu + \eta_t^2 \mu L\right).$$

Now, consider the second term on the right hand side of (4.9). We have,

$$\sum_{i=1}^{N} \mathbb{E}\left[\left\|\mathbf{x}_i^{t+1} - \bar{\mathbf{x}}^{t+1} - \mathbb{E}[\mathbf{x}_i^{t+1} - \bar{\mathbf{x}}^{t+1}]\right\|^2 |\mathcal{F}^t\right] =$$
$$\sum_{i=1}^{N} \mathbb{E}\left[\left\|\mathbf{x}_i^{t+1} - \mathbb{E}[\mathbf{x}_i^{t+1}] - (\bar{\mathbf{x}}^{t+1} - \mathbb{E}[\bar{\mathbf{x}}^{t+1}])\right\|^2 |\mathcal{F}^t\right]$$
$$= \eta_t^2 \sum_{i=1}^{N} \mathbb{E}\left[\left\|\mathbf{w}_i^t - \bar{\mathbf{w}}^t\right\|^2 |\mathcal{F}^t\right]$$
$$= \eta_t^2 \left(\sum_{i=1}^{N} \mathbb{E}\left[\left\|\mathbf{w}_i^t\right\|^2 |\mathcal{F}^t\right] - N\mathbb{E}\left[\left\|\bar{\mathbf{w}}^t\right\|^2 |\mathcal{F}^t\right]\right)$$
$$= \eta_t^2 \sum_{i=1}^{N} \mathbb{E}\left[\left\|\mathbf{w}_i^t\right\|^2 |\mathcal{F}^t\right] (1 - \frac{1}{N})$$
$$\leq (N-1)\eta_t^2 \sigma^2 + (1 - \frac{1}{N})\eta_t^2 c \sum_{i=1}^{N} \|\mathbf{g}_i^t\|^2,$$

where $\mathbf{w}_i^t$ are defined at the beginning of this section and $\bar{\mathbf{w}}^t := (\sum_{i=1}^{N} \mathbf{w}_i^t)/n$ and we used Lemma 33 in the third equation and the conditional independence of $\mathbf{w}_i^t$ to use $\mathbb{E}[\|\bar{\mathbf{w}}^t\|^2|\mathcal{F}^t] = (1/N^2)\sum_{i=1}^{N} \mathbb{E}[\|\mathbf{w}_i^t\|^2|\mathcal{F}^t]$ in the last equality. Taking full expectation of the two relations above with respect to $\mathcal{F}^t$ and combining them with (4.9) completes the proof. □

Lemma 32, bounds how much the consensus error grows at each iteration. Of course, when workers communicate, this error resets to zero and thus, we can calculate an upper bound for the consensus error, knowing the last iteration communication occurred and the step-size sequence. The following lemma takes care of that. Before stating the following lemma, let us define $G^t := \frac{1}{n}\sum_{i=1}^{N} \|\mathbf{g}_i^t\|^2$.

Lemma 34. *Let assumptions of Theorem 5 hold. Then,*

$$\mathbb{E}\left[\sum_{i=1}^{N}\|\mathbf{x}_i^t-\bar{\mathbf{x}}^t\|^2\right]\leq 12(N-1)\sum_{k=\tau(t)}^{t-1}\frac{c\mathbb{E}\left[G^k\right]+\sigma^2}{\mu^2(t+\beta)^2}. \tag{4.13}$$

Before proving Lemma 34, let us state and prove the following lemma.

Lemma 35. *Let $b\geq a>2$ be integers. Define $\Phi(a,b)=\prod_{i=a}^{b}\left(1-\frac{2}{i}\right)$. We then have $\Phi(a,b)\leq\left(\frac{a}{b+1}\right)^2$.*

Proof. Indeed,

$$\ln(\Phi(a,b))=\sum_{i=a}^{b}\ln\left(1-\frac{2}{i}\right)\leq\sum_{i=a}^{b}-\frac{2}{i}\leq-2\left[\ln(b+1)-\ln(a)\right].$$

where we used the inequality $\ln(1-x)\leq -x$ as well as the standard technique of viewing $\sum_{i=a}^{b}1/i$ as a Riemann sum for $\int_a^{b+1}1/x\,dx$ and observing that the Riemann sum overstates the integral. Exponentiating both sides now implies the lemma. □

Proof of Lemma 34. Define $a^k=\mathbb{E}\left[\sum_{i=1}^{N}\|\mathbf{x}_i^k-\bar{\mathbf{x}}^k\|^2\right]$ and $\Delta_k=(1-\eta_k\mu+\eta_k^2\mu L)$ for $k\geq 0$. By Lemma 32,

$$\begin{aligned}
a^t&\leq\Delta_{t-1}a^{t-1}+\eta_{t-1}^2(N-1)(\sigma^2+c\mathbb{E}[G^{t-1}])\\
&\leq\Delta_{t-1}(\Delta_{t-2}a^{t-2}+\eta_{t-2}^2(N-1)(\sigma^2+c\mathbb{E}[G^{t-2}]))+\eta_{t-1}^2(N-1)(\sigma^2+c\mathbb{E}[G^{t-1}])\\
&\leq\ldots\leq\prod_{k=\tau(t)}^{t-1}\Delta_k a^{\tau(t)}+(N-1)\sum_{k=\tau(t)}^{t-1}\eta_k^2(\sigma^2+c\mathbb{E}[G^k])\prod_{i=k+1}^{t-1}\Delta_i\\
&=(N-1)\sum_{k=\tau(t)}^{t-1}\eta_k^2(\sigma^2+c\mathbb{E}[G^k])\prod_{i=k+1}^{t-1}\Delta_i,
\end{aligned}$$

where we used $a^{\tau(t)}=0$ in the last equation. By the choice of stepsize and $\beta\geq 9\kappa$, we have

$$\Delta_k=1-\frac{3}{k+\beta}+\frac{9L}{\mu(k+\beta)^2}\leq 1-\frac{3}{k+\beta}+\frac{9\kappa}{(k+\beta)\beta}\leq 1-\frac{3}{k+\beta}+\frac{1}{(k+\beta)}=1-\frac{2}{k+\beta}.$$

Therefore, by Lemma 35,

$$a^t\leq(N-1)\sum_{k=\tau(t)}^{t-1}\frac{9(\sigma^2+c\mathbb{E}[G^k])}{\mu^2(k+\beta)^2}\frac{(k+\beta+1)^2}{(t+\beta)^2}\leq(N-1)\sum_{k=\tau(t)}^{t-1}\frac{12(\sigma^2+c\mathbb{E}[G^k])}{\mu^2(t+\beta)^2},$$

where we used $9(k+\beta+1)^2/(k+\beta)^2\leq 9(\beta+1)^2/\beta^2\leq 9(10/9)^2\leq 12$ since $\beta\geq 9\kappa\geq 9$. □

Variance. Our next lemma bounds $E[\|\tilde{\mathbf{g}}^t\|^2]$.

Lemma 36. *Under Assumption 5 we have,*

$$\mathbb{E}\left[\left\|\tilde{\mathbf{g}}^t\right\|^2\right] \leq \left(1+\frac{c}{N}\right)\mathbb{E}\left[G^t\right] + \frac{\sigma^2}{N}.$$

Proof. We have,

$$\mathbb{E}[\|\tilde{\mathbf{g}}^t\|^2|\mathcal{F}^t] = \mathbb{E}[\|\bar{\mathbf{g}}^t + \bar{\epsilon}^t\|^2|\mathcal{F}_t] = \|\bar{\mathbf{g}}^t\|^2 + \mathbb{E}[\|\bar{\mathbf{w}}^t\|^2|\mathcal{F}^t]$$
$$\leq \frac{1}{N}\sum_{i=1}^N \|\mathbf{g}_i^t\|^2 + \frac{1}{N^2}\sum_{i=1}^N(\sigma^2 + c\|\mathbf{g}_i^t\|^2),$$

where in the last inequality we used Lemma 33 and the conditional independency of $\mathbf{w}_i^t$ to decouple the noise terms. □

Now, we are ready to prove Theorem 5.

Proof of Theorem 5. Combining Equations Lemmas 31-36 and plugging $\eta_t = 3/(\mu(t+\beta))$ we obtain

$$\xi^{t+1} \leq \xi^t(1-\mu\eta_t) + \frac{18L^2}{\mu^3(t+\beta)^3}\sum_{k=\tau(t)}^{t-1}\left(c\mathbb{E}[G^k] + \sigma^2\right)$$
$$+ \frac{9L}{2\mu^2(t+\beta)^2}\left(\left(1+\frac{c}{N}\right)\mathbb{E}[G^t] + \frac{\sigma^2}{N}\right) - \frac{3}{2\mu(t+\beta)}\mathbb{E}[G^t].$$

Let us multiply both sides of relation above by $(t+\beta)^2$ and use the following inequality

$$(1-\mu\eta_t)(t+\beta)^2 = \left(1-\frac{2}{t+\beta}\right)(t+\beta)^2 = (t+\beta)^2 - 2(t+\beta) < (t+\beta-1)^2,$$

to obtain,

$$\xi^{t+1}(t+\beta)^2 \leq \xi^t(t+\beta-1)^2 + \frac{9L\sigma^2}{2\mu^2 N} +$$
$$\frac{18L^2}{\mu^3(t+\beta)}\sum_{k=\tau(t)}^{t-1}\left(c\mathbb{E}[G^k] + \sigma^2\right) + \left(\frac{9L}{2\mu^2}\left(1+\frac{c}{N}\right) - \frac{3(t+\beta)}{2\mu}\right)\mathbb{E}[G^t].$$

Summing relation above for $t = \tau_i, \ldots, \tau_{i+1} - 1$, where $\tau_i, \tau_{i+1} \in \mathcal{I}$ are two consecutive

communication times, implies,

$$\xi^{\tau_{i+1}}(\tau_{i+1}+\beta-1)^2 \le \xi^{\tau_i}(\tau_i+\beta-1)^2 + \frac{9L\sigma^2}{2\mu^2 N}(\tau_{i+1}-\tau_i) + \frac{18L^2\sigma^2}{\mu^3}\sum_{t=\tau_i}^{\tau_{i+1}-1}\frac{t-\tau_i}{t+\beta} + \sum_{t=\tau_i}^{\tau_{i+1}-1}\mathbb{E}[G^t]\left(\sum_{k=t+1}^{\tau_{i+1}-1}\frac{18L^2c}{\mu^3(k+\beta)} + \frac{9L}{2\mu^2}\left(1+\frac{c}{N}\right) - \frac{3(t+\beta)}{2\mu}\right).$$

Each of the coefficients of $\mathbb{E}[G^t]$ in above can be bounded by,

$$\begin{aligned}\sum_{k=t+1}^{\tau_{i+1}-1}\frac{18L^2c}{\mu^3(k+\beta)} + \frac{9L}{2\mu^2}\left(1+\frac{c}{N}\right) - \frac{3(t+\beta)}{2\mu} &\le \frac{18L^2c}{\mu^3}\ln\left(\frac{\tau_{i+1}+\beta-1}{\tau_i+\beta}\right) + \frac{9L}{2\mu^2}\left(1+\frac{c}{N}\right) - \frac{3\tau_i+\beta}{2\mu}\\ &= \frac{3}{2\mu}\left(12\kappa^2 c\ln\left(1+\frac{H_i-1}{\tau_i+\beta}\right) + 3\kappa\left(1+\frac{c}{N}\right) - (\tau_i+\beta)\right)\\ &\le 0,\end{aligned}$$

where we used $\sum_{k=t_1+1}^{t_2} 1/k \le \int_{t_1}^{t_2} dx/x = \ln(t_2/t_1)$ in the first inequality and the last inequality comes from the assumption of the theorem. Now that the coefficients of $\mathbb{E}[G^k]$ are non-positive, we can simply ignore them and obtain,

$$\xi^{\tau_{i+1}}(\tau_{i+1}+\beta-1)^2 \le \xi^{\tau_i}(\tau_i+\beta-1)^2 + \frac{9L\sigma^2}{2\mu^2 N}(\tau_{i+1}-\tau_i) + \frac{18L^2\sigma^2}{\mu^3}\sum_{t=\tau_i}^{\tau_{i+1}-1}\frac{t-\tau_i}{t+\beta}.$$

Recursing relation above for $i = 0, \ldots, R-1$ implies,

$$\xi^T(T+\beta-1)^2 \le \xi^0(\beta-1)^2 + \frac{9L\sigma^2}{2\mu^2 N}T + \frac{18L^2\sigma^2}{\mu^3}\sum_{t=0}^{T-1}\frac{t-\tau(t)}{t+\beta}.$$

Dividing both sides by $(T+\beta-1)^2$ concludes the proof. □

Proof of Theorem 6. We have,

$$\tau_j = \tau_0 + \sum_{i=0}^{j-1} H_i = a\frac{j(j+1)}{2}, \qquad j = 0, \ldots, k-1.$$

Hence,

$$\begin{aligned}1 + \frac{H_0-1}{\tau_0+\beta} &= 1 + \frac{a-1}{\beta} \le 1 + \frac{2T}{9\kappa R^2} \le 1 + \frac{T}{4\kappa R^2},\\ 1 + \frac{H_i-1}{\tau_i+\beta} &\le 1 + \frac{a(i+1)}{\frac{ai(i+1)}{2}} \le 3, \qquad i \ge 1.\end{aligned}$$

Thus, $12\kappa^2 c \ln(1+\frac{H_i-1}{\tau_i+\beta})+3\kappa(1+\frac{c}{N})-(\tau_i+\beta) \leq 0, i=0,\ldots,R-1$ and we can use Theorem 5. Moreover,

$$\begin{aligned}\sum_{t=0}^{T-1}\frac{t-\tau(t)}{t+\beta} &\leq \sum_{j=0}^{R-1}\sum_{i=1}^{H_j-1}\frac{i}{\tau_j+i+\beta} \leq H_0+\sum_{j=1}^{R-1}\sum_{i=1}^{H_j-1}\frac{i}{\tau_j+1+\beta} \\ &= a+\sum_{j=1}^{R-1}\frac{H_j(H_j-1)}{2(\tau_j+1+\beta)} = a+\sum_{j=1}^{R-1}\frac{a(j+1)(a(j+1)-1)}{aj(j+1)+2(1+\beta)} \\ &\leq a+\sum_{j=1}^{R-1}\frac{a^2(j+1)^2}{aj(j+1)} \leq 2aR.\end{aligned}$$

Plugging the values of R and a implies,

$$\sum_{t=0}^{T-1}\frac{t-\tau(t)}{t+\beta} \leq 2aR \leq 2(\frac{2T}{R^2}+1)R = \frac{4T}{R}+2R \leq \frac{4T}{R}+\frac{4T}{R} = \frac{8T}{R},$$

where we used $R \leq \sqrt{2T}$ in the last inequality. Using the relation above together with Theorem 5 concludes the proof. □

4.4 Convergence results for one-shot averaging

The previous literature has shown OSA achieves asymptotic linear speed-up under some restrictive assumptions. For instance, Dieuleveut and Patel [2019] show this for three times continuously differentiable functions with second and third uniformly bounded derivatives. Similarly, Godichon-Baggioni and Saadane [2020] requires the objective function to be strongly convex, twice continuously differentiable almost everywhere, with a bounded Hessian everywhere and gradients satisfying the following condition for some constant C_m and all $\mathbf{x} \in \mathbb{R}^d$,

$$\left\|\nabla f(\mathbf{x}) - \nabla^2 f(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*)\right\| \leq C_m \|\mathbf{x} - \mathbf{x}^*\|^2.$$

This inequality is similar to the assumption from Dieuleveut and Patel [2019] of uniformly bounded third derivatives. In the following theorem, we relax these assumptions and show that OSA achieves linear speed-up under considerably milder assumptions.

Before proceeding, let us define the step-size sequence $\{\theta_t\}$ as

$$\theta_t = \begin{cases} \frac{1}{L}, & \text{for } t = 0, \ldots, t_0 - 1, \\ \frac{2t}{\mu(t+1)^2}, & \text{for } t \geq t_0, \end{cases} \tag{4.14}$$

where $t_0 = \lfloor 2L/\mu \rfloor$. Notice that $\theta_t \leq 1/L$ for all t.

Theorem 7. *Under Assumptions 4 (smoothness), 6 (PL condition), 7 (twice differentiability at the optimum) and 9 (sub-Gaussian noise) and with step-size sequence $\{\eta_t\} = \{\theta_t\}$ defined in (4.14), we have for $T \geq t_0$,*

$$\mathbb{E}\left[\left\|\bar{\mathbf{x}}^T - \mathbf{x}^*\right\|^2\right] \leq \frac{4\sigma^2}{3\mu^2 NT} + o\left(\frac{1}{T}\right).$$

We are thus able to relax the conditions from the earlier literature, which required everywhere or almost everywhere higher derivatives with uniform bounds on third derivatives to merely twice differentiability at a single point. As a bonus, we also replace strong convexity with the PL condition.

The main difference between Theorem 7 and Corollary 6 is that Theorem 7 shows a linear speed-up with only one communication round but with slightly more restrictive assumptions such as sub-Gaussian noise model and twice-differentiable objective function at the optimal point. On the other hand, our results for OSA only require the PL-condition instead of strong convexity.

4.4.1 Poorf of results for OSA

In this section we prove Theorem 7 for one-shot averaging. The main idea is to use second order approximation for gradients at any point with respect to the minimizer and show that the residual errors are *insignificant*, using concentration results from Karimi et al. [2016].

Define $\mathbf{v}(\mathbf{y}, \mathbf{x}) := \nabla f(\mathbf{y}) - \big(\nabla f(\mathbf{x}) + \nabla^2 f(\mathbf{x})(\mathbf{y} - \mathbf{x})\big)$ and $\mathbf{v}_i^t = \mathbf{v}(\mathbf{x}_i^t, \mathbf{x}^*)$.

Lemma 37. *Let Assumption 7 hold. Then* $|[\mathbf{v}(\mathbf{x}, \mathbf{x}^*)]_i| = o(\|\mathbf{x} - \mathbf{x}^*\|)$ *for* $i = 1, \ldots, d$.

Proof. Denote $h_i(\mathbf{x}) = [\nabla f(\mathbf{x})]_i$. Then by Assumption 7, h_i is continuously differentiable over an open set containing $\mathbf{x}^*$. Thus,

$$h_i(\mathbf{x}) = h_i(\mathbf{x}^*) + \nabla h_i(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) + o(\|\mathbf{x} - \mathbf{x}^*\|) = \nabla h_i(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) + o(\|\mathbf{x} - \mathbf{x}^*\|).$$

Therefore,

$$[\mathbf{v}(\mathbf{x}, \mathbf{x}^*)]_i = h_i(\mathbf{x}) - \sum_{j=1}^{d} \frac{\partial^2 f}{\partial x_i \partial x_j}(\mathbf{x}^*)[\mathbf{x} - \mathbf{x}^*]_j = h_i(\mathbf{x}) - \nabla h_i(\mathbf{x}^*)^\top (\mathbf{x} - \mathbf{x}^*) = o(\|\mathbf{x} - \mathbf{x}^*\|).$$

□

Let us define $u(r) := \max_{\|\mathbf{x} - \mathbf{x}^*\| \le r} \|\mathbf{v}(\mathbf{x}, \mathbf{x}^*)\|$. We have $u(r) = o(r)$.

Theorem 8 (Karimi et al. [2016], Theorem 1)**.** *Under Assumptions 4 and 6, the following inequality, known as the quadratic growth (QG) condition holds:*

$$\|\mathbf{x} - \mathbf{x}^*\|^2 \le \frac{2}{\mu}(f(\mathbf{x}) - f^*).$$

Lemma 38. *Under Assumptions 4, 6 and 7 we have,*

$$\nabla^2 f(\mathbf{x}^*) \succeq \mu.$$

Proof. The result is established by using the *linear approximation theorem* on a sequence of points converging to x^* on a line, continuity of Hessian as well as the quadratic growth from Theorem 8. Similar approach can be found in the proof of Theorem 2.26 Beck [2014]. □

Next, we state a Theorem from Madden et al. [2020] which we will use frequently in the rest of our results.

Theorem 9 (Madden et al. [2020], Theorems 4 and 13)**.** *Under Assumptions 4, 6 and 9, SGD with step-size sequence* $\{\eta_t\} = \{\theta_t\}$ *defined in* (4.14)*, constructs a sequence of* $\{\mathbf{x}^t\}$ *such that there exist* $C_1, C_2 > 0$ *such that for* $t \geq t_0$*,*

$$\mathbb{E}[f(\mathbf{x}^t)] - f^* = C_1 \frac{L\sigma^2}{\mu^2 t},$$

and w.p. $\geq 1 - \delta$ *for all* $\delta \in (0, 1/e)$*,*

$$f(\mathbf{x}^t) - f^* \leq C_2 \frac{L\sigma^2 \log(e/\delta)}{\mu^2 t}.$$

Lemma 39. *Under Assumptions 4 and 7 we have,*

$$\|\mathbf{v}(\mathbf{x}, \mathbf{x}^*)\| \leq 2L\|\mathbf{x} - \mathbf{x}^*\|. \tag{4.15}$$

Proof. We have,

$$\begin{aligned}
\|\mathbf{v}(\mathbf{x}, \mathbf{x}^*)\| &= \|\nabla f(\mathbf{x}) - \nabla^2 f(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*)\| \\
&\leq \|\nabla f(\mathbf{x})\| + \|\nabla^2 f(\mathbf{x}^*)(\mathbf{x} - \mathbf{x}^*)\| \\
&\leq L\|\mathbf{x} - \mathbf{x}^*\| + \|\nabla^2 f(\mathbf{x}^*)\|_2 \|\mathbf{x} - \mathbf{x}^*\| \\
&\leq 2L\|\mathbf{x} - \mathbf{x}^*\|,
\end{aligned}$$

where we used $\|\nabla^2 f(\mathbf{x}^*)\| \leq L$ in the last inequality. □

The following lemma is the key result we need to show the asymptotic performance of OSA.

Lemma 40. *Under Assumptions 4, 6, 7 and 9 and steps-size sequence* $\{\eta_t\} = \{\theta_t\}$ *defined in* (4.14)*, we have*

1. $\mathbb{E}[\|\mathbf{v}_i^t\|^2] = o(\frac{1}{t})$,
2. $\mathbb{E}[\|\mathbf{v}_i^t\| \|\mathbf{x}_i^t - \mathbf{x}^*\|] = o(\frac{1}{t})$.

Proof. Let us define $u(r) := \max_{\|\mathbf{x}-\mathbf{x}^*\|\leq r} \|\mathbf{v}(\mathbf{x},\mathbf{x}^*)\|$. By Lemma 37 we have $u(r) = o(r)$. Also define random variable $r_i^t = \|\mathbf{x}_i^t - \mathbf{x}^*\|$.

Since $u(r) = o(r)$, for any $\epsilon > 0$ there exists $s > 0$ such that for $r \leq s$, $u(r) \leq \sqrt{\epsilon} r$ or $u(r)^2 \leq \epsilon r^2$. We have,

$$\begin{aligned}
\mathbb{E}[\|\mathbf{v}_i^t\|^2] &= \mathbb{E}_{\mathbf{x}_i^t}[\|\mathbf{v}(\mathbf{x}_i^t,\mathbf{x}^*)\|^2] \leq \mathbb{E}_{r_i^t}[u(r_i^t)^2] \\
&= \int_0^\infty u(r)^2 p_{r_i^t}(r) dr \\
&= \int_0^s u(r)^2 p_{r_i^t}(r) dr + \int_s^\infty u(r)^2 p_{r_i^t}(r) dr \\
&\leq \epsilon \int_0^s r^2 p_{r_i^t}(r) dr + 4L^2 \int_s^\infty r^2 p_{r_i^t}(r) dr \\
&= \epsilon \mathbb{E}[(r_i^t)^2] + (4L^2 - \epsilon) \int_s^\infty r^2 p_{r_i^t}(r) dr, \qquad (4.16)
\end{aligned}$$

where p_X denotes the Probability Density Function (PDF) for random variable X and we used $u(r) \leq 2Lr$ from (4.15).

Without loss of generality, we assume $t \geq t_0$ for the rest of the proof. By Theorems 8 and 9 we have,

$$\mathbb{E}\left[(r_i^t)^2\right] = \mathbb{E}\left[\left\|\mathbf{x}_i^t - \mathbf{x}^*\right\|^2\right] \leq \frac{2}{\mu}\mathbb{E}[f(\mathbf{x}_i^t) - f^*] \leq \frac{2C_1 L\sigma^2}{\mu^2 t} = \mathcal{O}\left(\frac{1}{t}\right).$$

Moreover, define $J_t(\delta) := C_2 L\sigma^2 \log(e/\delta)/(\mu^2 t)$. Then,

$$\Pr\left((r_i^t)^2 \leq \frac{2J_t(\delta)}{\mu}\right) \geq \Pr\left(f(\mathbf{x}_i^t) - f^* \leq J_t(\delta)\right) \geq 1 - \delta, \qquad \text{for } \delta \in (0, 1/e), \qquad (4.17)$$

or,

$$F^{-1}_{(r_i^t)^2}(1-\delta) \leq \frac{2J_t(\delta)}{\mu}, \qquad \text{for } \delta \in (0, 1/e), \qquad (4.18)$$

where F_X denotes the Cumulative Distribution Function (CDF) for random variable X.

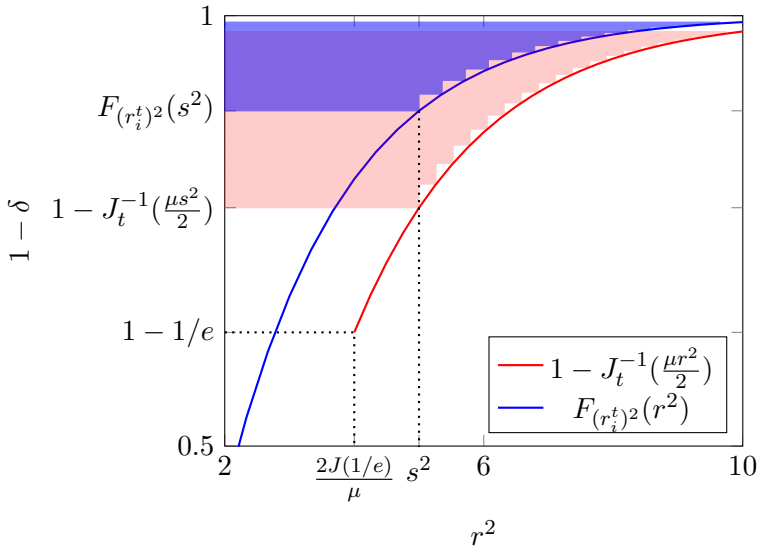


Figure 4·1: Illustration of integrals in (4.19)

Since $\lim_{t\to\infty} J_t(\delta) = 0$, $\exists t_1 \geq t_0$ such that for $t \geq t_1$, $J_t^{-1}(\mu s^2/2) \in (0, 1/e)$. It follows that,

$$\begin{aligned}
\int_s^\infty r^2 p_{r_i^t}(r)dr &= \int_{s^2}^\infty r^2 p_{(r_i^t)^2}(r^2)dr^2 = \int_{s^2}^\infty r^2 dF_{(r_i^t)^2}(r^2) \\
&= \int_{F_{(r_i^t)^2}(s^2)}^{1} F^{-1}_{(r_i^t)^2}(x)dx = \int_{1-F_{(r_i^t)^2}(s^2)}^{0} -F^{-1}_{(r_i^t)^2}(1-\delta)d\delta \\
&= \int_0^{1-F_{(r_i^t)^2}(s^2)} F^{-1}_{(r_i^t)^2}(1-\delta)d\delta \\
&\leq \frac{2}{\mu} \int_0^{1-F_{(r_i^t)^2}(s^2)} J_t(\delta)d\delta \leq \frac{2}{\mu} \int_0^{J_t^{-1}(\frac{\mu s^2}{2})} J_t(\delta)d\delta. \qquad (4.19)
\end{aligned}$$

In the equation above, we switched from Probability Density Function (PDF) $p_{r_i^t}$ to $p_{(r_i^t)^2}$ in the first equality. In the next equality we used $p_X = dF_X/dX$ that holds for any continuous random variable X. In third equality, we simply changed variable to $x = F_{(r_i^t)^2(r^2)}$ and without loss of generality we define $F_X^{-1}(y) := \inf\{x|F_X(x) \geq y\}$. In the next equation, again, we simply changed variable to $\delta = 1 - x$. Finally, in the last two inequalities, we used (4.18) and a direct result of (4.17), $1 - F_{(r_i^t)^2}(s^2) \leq J_t^{-1}(\frac{\mu s^2}{2})$ (see Figure 4·1).

By Lemma 41, $\exists t_2 \geq t_1$ such that for $t \geq t_2$, $\int_0^{J_t^{-1}(\mu s^2/2)} J_t(\delta)d\delta \leq \epsilon B_1/t$, where $B_1 :=$

$2C_2L\sigma^2 e/\mu^2$. Combining with (4.16) we obtain,

$$\mathbb{E}[\|\mathbf{v}_i^t\|^2] \leq \frac{\epsilon}{t}\left(\frac{2C_1L\sigma^2}{\mu^2} + \frac{16C_2L^3\sigma^2 e}{\mu^3}\right), \qquad t \geq t_2.$$

Next, we show $\mathbb{E}[\|\mathbf{v}_i^t\|\|\mathbf{x}_i^t - \mathbf{x}^*\|] = o(1/t)$. Since $u(r) = o(r)$, for any $\epsilon > 0$, there exists $s' > 0$ such that for $r \leq s'$, $u(r) \leq \epsilon r$. Then,

$$\begin{aligned}
\mathbb{E}[\|\mathbf{v}_i^t\|\|\mathbf{x}_i^t - \mathbf{x}^*\|] &\leq \mathbb{E}[u(r_i^t)r_i^t] \\
&= \int_0^{s'} u(r)rp_{r_i^t}(r)dr + \int_{s'}^{\infty} u(r)rp_{r_i^t}(r)dr \\
&\leq \epsilon \int_0^{s'} r^2 p_{r_i^t}(r) + 4L^2 \int_{s'}^{\infty} r^2 p_{r_i^t}(r)dr.
\end{aligned}$$

Following the same steps from the first part of this proof, we obtain $\exists t_3 > 0$ such that,

$$\mathbb{E}[\|\mathbf{v}_i^t\|\|\mathbf{x}_i^t - \mathbf{x}^*\|] \leq \frac{\epsilon}{t}\left(\frac{2C_1L\sigma^2}{\mu^2} + \frac{16C_2L^3\sigma^2 e}{\mu^3}\right), \qquad t \geq t_3.$$

Since we could pick ϵ arbitrarily small, we showed that $\mathbb{E}[\|\mathbf{v}_i^t\|^2] = o(1/t)$ and $\mathbb{E}[\|\mathbf{v}_i^t\|\|\mathbf{x}_i^t - \mathbf{x}^*\|] = o(1/t)$. □

Lemma 41. *Let $q_t : (0, 1/e) \to \mathbb{R}_+$ be defined as $q_t(\delta) = a_1 \log(e/\delta)/t$ for some $a_1 > 0$ and $\forall t \geq 1$. Suppose $y \in range(q_t)$ for $t \geq t_1$, then for any $\epsilon > 0$, there exists $t_2 \geq t_1$ such that for any $t \geq t_2$,*

$$\int_0^{q_t^{-1}(y)} q_t(\delta)d\delta \leq \frac{B\epsilon}{t},$$

where $B = 2a_1 e$.

Proof. Define x_t such that $q_t(x_t) = y$. Then,

$$\frac{a_1 \log(e/x_t)}{t} = y \iff \log(\frac{e}{x_t}) = \frac{yt}{a_1} \iff x_t = \exp(1 - \frac{yt}{a_1}). \tag{4.20}$$

Moreover,

$$\begin{aligned}\int_0^{x_t} q_t(\delta)d\delta &= \frac{a_1}{t}\int_0^{x_t}\log(\frac{e}{\delta})d\delta \\ &= \frac{a_1}{t}\left(x_t - x_t(\log(x_t)-1)\right) \\ &= \frac{a_1}{t}\left(x_t + x_t(\frac{yt}{a_1})\right) = x_t(y+\frac{a_1}{t}).,\end{aligned}$$

where we used (4.20) in third equality. First, we note that for $t \geq a_1/y$, we have $y + a_1/t \leq 2y$. Next, we show that for t large enough, $x_t \leq B\epsilon/(2yt)$ for some $B > 0$. We have $\lim_{s\to\infty}\exp(s)/s = \infty$. Therefore $\exists s_0 \geq 1$ such that for $s \geq s_0$, $\exp(s)/s \geq 1/\epsilon$. Thus for $t \geq s_0 a_1/y$ we have,

$$\begin{aligned}&\exp(\frac{ty}{a_1}) \geq \frac{ty}{a_1\epsilon}, \\ \Rightarrow &x_t = \exp(1-\frac{yt}{a_1}) \leq e(\frac{a_1\epsilon}{ty}) = \frac{B\epsilon}{2yt},\end{aligned}$$

where $B := 2a_1 e$. Therefore, for $t \geq t_2 := \max\{s_0 a_1/y, t_1\}$ we have $\int_0^{x_t} q_t(\delta)d\delta \leq 2x_t y \leq B\epsilon/t$. □

Now, we are ready to derive the one-step progress in the following lemma.

Lemma 42. *Under Assumptions 4, 6, 7 and 9 and steps-size sequence $\{\eta_t\} = \{\theta_t\}$ defined in (4.14), we have*

$$\mathbb{E}[\|\bar{\mathbf{x}}^{t+1} - \mathbf{x}^*\|^2] \leq (1-\eta_t\mu)^2\mathbb{E}[\|\bar{\mathbf{x}}^t - \mathbf{x}^*\|^2] + \frac{\eta_t^2\sigma^2}{N} + o\left(\frac{1}{t^2}\right). \tag{4.21}$$

Proof. Let us define $\mathbf{A} = \nabla^2 f(\mathbf{x}^*)$. By definition,

$$\nabla f(\mathbf{x}_i^t) = \mathbf{A}(\mathbf{x}_i^t - \mathbf{x}^*) + \mathbf{v}_i^t. \tag{4.22}$$

Plugging (4.22) in SGD process and averaging over all i we obtain,

$$\begin{aligned}\bar{\mathbf{x}}^{t+1} &= \bar{\mathbf{x}}^t - \frac{\eta_t}{N}\sum_{i=1}^N \hat{\mathbf{g}}_i^t = \bar{\mathbf{x}}^t - \frac{\eta_t}{N}\sum_{i=1}^N\left(\nabla f(\mathbf{x}_i^t) + \mathbf{w}_i^t\right) \\ &= \bar{\mathbf{x}}^t - \frac{\eta_t}{N}\sum_{i=1}^N\left(\mathbf{A}(\mathbf{x}_i^t - \mathbf{x}^*) + \mathbf{v}_i^t + \mathbf{w}_i^t\right) = \bar{\mathbf{x}}^t - \eta_t\mathbf{A}(\bar{\mathbf{x}}^t - \mathbf{x}^*) + \frac{\eta_t}{N}\sum_{i=1}^N(\mathbf{v}_i^t + \mathbf{w}_i^t).\end{aligned}$$

Thus,

$$\begin{aligned}\mathbb{E}[\|\bar{\mathbf{x}}^{t+1}-\mathbf{x}^*\|^2|\mathcal{F}_t] &= \mathbb{E}[\|(I-\eta_t\mathbf{A})(\bar{\mathbf{x}}^t-\mathbf{x}^*)+\frac{\eta_t}{N}\sum_{i=1}^N(\mathbf{v}_i^t+\mathbf{w}_i^t)\|^2|\mathcal{F}_t]\\ &= \mathbb{E}[\|(I-\eta_t\mathbf{A})(\bar{\mathbf{x}}^t-\mathbf{x}^*)+\frac{\eta_t}{N}\sum_{i=1}^N\mathbf{v}_i^t\|^2|\mathcal{F}_t]+\mathbb{E}[\|\frac{\eta_t}{N}\sum_{i=1}^N\mathbf{w}_i^t\|^2|\mathcal{F}_t]\\ &\leq \mathbb{E}[\|(I-\eta_t\mathbf{A})(\bar{\mathbf{x}}^t-\mathbf{x}^*)+\frac{\eta_t}{N}\sum_{i=1}^N\mathbf{v}_i^t\|^2|\mathcal{F}_t]+\frac{\eta_t^2\sigma^2}{N},\end{aligned}$$

where $\mathcal{F}^t := \{\mathbf{x}_i^k, \hat{\mathbf{g}}_i^k | 1\leq i\leq N, 0\leq k\leq t-1\}\cup\{\mathbf{x}_i^t|1\leq i\leq N\}$. Taking full expectation with respect to $\mathcal{F}_t$ yields,

$$\begin{aligned}\mathbb{E}[\|\bar{\mathbf{x}}^{t+1}-\mathbf{x}^*\|^2] &\leq \mathbb{E}[\|(I-\eta_t\mathbf{A})(\bar{\mathbf{x}}^t-\mathbf{x}^*)+\frac{\eta_t}{N}\sum_{i=1}^N\mathbf{v}_i^t\|^2]+\frac{\eta_t^2\sigma^2}{N}\\ &= \mathbb{E}[\|(I-\eta_t\mathbf{A})(\bar{\mathbf{x}}^t-\mathbf{x}^*)\|^2]+\frac{\eta_t^2\sigma^2}{N}\\ &\quad+\underbrace{\mathbb{E}[\|\frac{\eta_t}{N}\sum_{i=1}^N\mathbf{v}_i^t\|^2]}_{T_2}+\underbrace{\mathbb{E}[\|(I-\eta_t\mathbf{A})(\bar{\mathbf{x}}^t-\mathbf{x}^*)\|\|\frac{\eta_t}{N}\sum_{i=1}^N\mathbf{v}_i^t\|]}_{T_3}. \qquad (4.23)\end{aligned}$$

Next we bound T_2 and T_3. Using Lemma 40 we have,

$$\mathbb{E}[\|\frac{\eta_t}{N}\sum_{i=1}^N\mathbf{v}_i^t\|^2]\leq\frac{\eta_t^2}{N}\sum_{i=1}^N\mathbb{E}[\|\mathbf{v}_i^t\|^2]\leq\frac{4}{\mu^2t^2}o\left(\frac{1}{t}\right)=o\left(\frac{1}{t^3}\right). \qquad (4.24)$$

Moreover, by Lemma 38 and f being L-smooth we have $\mu\preceq\mathbf{A}\preceq L$. It follows

$$1-\eta_tL\preceq I-\eta_t\mathbf{A}\preceq 1-\eta_t\mu.$$

Since $\eta_t\leq 1/L$ and $I-\eta_t\mathbf{A}$ is symmetric, we have $\|I-\eta_t\mathbf{A}\|\leq 1-\eta_t\mu\leq 1$. Then,

$$\|(I-\eta_t\mathbf{A})(\bar{\mathbf{x}}^t-\mathbf{x}^*)\|\leq\|I-\eta_t\mathbf{A}\|\|\bar{\mathbf{x}}^t-\mathbf{x}^*\|\leq\|\bar{\mathbf{x}}^t-\mathbf{x}^*\|.$$

Thus,

$$\begin{aligned}
\mathbb{E}[\|(I-\eta_t\mathbf{A})(\bar{\mathbf{x}}^t-\mathbf{x}^*)\|\|\frac{\eta_t}{N}\sum_{i=1}^N\mathbf{v}_i^t\|] &\le \mathbb{E}[\|\bar{\mathbf{x}}^t-\mathbf{x}^*\|\|\frac{\eta_t}{N}\sum_{i=1}^N\mathbf{v}_i^t\|] \\
&\le \mathbb{E}\left[\left(\frac{1}{N}\sum_{i=1}^N\|\mathbf{x}_i^t-\mathbf{x}^*\|\right)\left(\frac{\eta_t}{N}\sum_{i=1}^N\|\mathbf{v}_i^t\|\right)\right] \\
&\le \frac{\eta_t}{N^2}\sum_{i=1}^N\mathbb{E}[\|\mathbf{x}_i^t-\mathbf{x}^*\|\|\mathbf{v}_i^t\|] + \frac{\eta_t}{N^2}\sum_{i\neq j}\mathbb{E}[\|\mathbf{x}_j^t-\mathbf{x}^*\|\|\mathbf{v}_i^t\|] \\
&= \frac{\eta_t}{N^2}\sum_{i=1}^N\mathbb{E}[\|\mathbf{x}_i^t-\mathbf{x}^*\|\|\mathbf{v}_i^t\|] \\
&+ \frac{\eta_t}{N^2}\sum_{i\neq j}\mathbb{E}[\|\mathbf{x}_j^t-\mathbf{x}^*\|]\mathbb{E}[\|\mathbf{v}_i^t\|] \\
&\le \frac{2}{\mu t}o\left(\frac{1}{t}\right)+\frac{2}{\mu t}o\left(\frac{1}{\sqrt{t}}\right)o\left(\frac{1}{\sqrt{t}}\right)=o\left(\frac{1}{t^2}\right), \qquad (4.25)
\end{aligned}$$

where we used $|E[X]| \le \sqrt{E[X^2]}$ for random variables $\|\mathbf{x}_j^t-\mathbf{x}^*\|$ and $\mathbf{v}_i^t$ and Lemma 40 in last equation above. plugging (4.24) and (4.25) in (4.23) we obtain the desired result. □

Now we are ready to present the proof of Theorem 7.

Proof of Theorem 7. Denote $\psi^t := \mathbb{E}[\|\bar{\mathbf{x}}^t-\mathbf{x}^*\|]$ for $t \ge 0$. By Lemma 42 we can write,

$$\psi^{t+1} \le \psi^t(1-\eta_t\mu)^2 + \frac{\eta_t^2\sigma^2}{N} + \nu^t,$$

where $\nu^t \ge 0$ and $\nu^t = o(1/t^2)$. It follows,

$$\psi^k \le \underbrace{\psi^0\prod_{t=0}^{k-1}(1-\eta_t\mu)^2}_{S_1} + \underbrace{\sum_{t=0}^{k-1}\frac{\eta_t^2\sigma^2}{N}\prod_{l=t+1}^{k-1}(1-\eta_l\mu)^2}_{S_2} + \underbrace{\sum_{t=0}^{k-1}\nu^t\prod_{l=t+1}^{k-1}(1-\eta_l\mu)^2}_{S_3}. \qquad \forall k \ge t_0. \tag{4.26}$$

Next, we will bound each of the terms S_1, S_2, and S_3. Before that, we note that for $t \ge t_0$,

$$1-\eta_t\mu = 1-\frac{2t}{(t+1)^2} \le 1-\frac{2}{t}.$$

Therefore, for $t_2 > t_1 \geq t_0$ we have,

$$\prod_{l=t_1}^{t_2-1}(1-\eta_l\mu) \leq \prod_{l=t_1}^{t_2-1}(1-\frac{2}{l}) = \exp\left(\sum_{l=t_1}^{t_2-1}\log(1-\frac{2}{l})\right)$$
$$\leq \exp\left(\sum_{l=t_1}^{t_2-1}\frac{-2}{l}\right) \leq \exp\left(2\log(t_1)-2\log(t_2)\right) = \left(\frac{t_1}{t_2}\right)^2. \quad (4.27)$$

Now we have the tools we need to bound S_1, S_2, and S_3. we have,

$$S_1 = \|\bar{\mathbf{x}}^0-\mathbf{x}^*\|^2\prod_{t=0}^{t_0-1}(1-\eta_t\mu)^2\prod_{t=t_0}^{k-1}(1-\eta_t\mu)^2 \leq (1-\frac{\mu}{L})^{2t_0}\left(\frac{t_0}{k}\right)^4\|\bar{\mathbf{x}}^0-\mathbf{x}^*\|^2. \quad (4.28)$$

$$\begin{aligned}
S_2 &= \sum_{t=0}^{t_0-1}\frac{\eta_t^2\sigma^2}{N}\prod_{l=t+1}^{t_0-1}(1-\eta_l\mu)^2\prod_{l=t_0}^{k-1}(1-\eta_l\mu)^2 + \sum_{t=t_0}^{k-1}\frac{\eta_t^2\sigma^2}{N}\prod_{l=t+1}^{k-1}(1-\eta_l\mu)^2 \\
&\leq \frac{\sigma^2}{N}\left[\sum_{t=0}^{t_0-1}\frac{1}{L^2}(1-\frac{\mu}{L})^{2(t_0-1-t)}\left(\frac{t_0}{k}\right)^4 + \sum_{t=t_0}^{k-1}\left(\frac{2t}{\mu(t+1)^2}\right)^2\left(\frac{t+1}{k}\right)^4\right] \\
&= \frac{\sigma^2}{N}\left[\frac{t_0^4}{L^2k^4}\sum_{t=0}^{t_0-1}(1-\frac{\mu}{L})^{2t} + \frac{4}{\mu^2k^4}\sum_{t=t_0}^{k-1}t^2\right] \\
&\leq \frac{\sigma^2}{N}\left[\frac{t_0^4}{L^2k^4}\sum_{t=0}^{\infty}(1-\frac{\mu}{L})^{2t} + \frac{4}{\mu^2k^4}\sum_{t=1}^{k-1}t^2\right] \\
&= \frac{\sigma^2}{N}\left[\frac{t_0^4}{L^2k^4(1-(1-\frac{\mu}{L})^2)} + \frac{2k(k-1)(2k-1)}{3\mu^2k^4}\right] \\
&\leq \frac{\sigma^2}{N}\left[\frac{t_0^4}{L\mu k^4} + \frac{4}{3\mu^2k}\right] = \frac{4\sigma^2}{3N\mu^2k}\left[1+\frac{3\mu t_0^4}{4Lk^3}\right]. \quad (4.29)
\end{aligned}$$

Next, we show $S_3 = o(1/k)$. Since $\nu^t = o(1/t^2)$, without loss of generality, we can assume there exists $B_1, B_2 > 0$ such that for any $\epsilon > 0$, there exists $k_1 \geq t_0$ such that,

$$\nu^t \leq \begin{cases} \frac{B_1}{(t+1)^2}, & t \geq 0, \\ \frac{\epsilon B_2}{(t+1)^2}, & t \geq k_1. \end{cases}$$

It follows,

$$
\begin{aligned}
S_3 &\leq \sum_{t=0}^{t_0-1}(1-\frac{\mu}{L})^{2(t_0-1-t)}\left(\frac{t_0}{k}\right)^4\frac{B_1}{(t+1)^2}+\sum_{t=t_0}^{k_1-1}\left(\frac{t+1}{k}\right)^4\frac{B_1}{(t+1)^2}+\sum_{t=k_1}^{k-1}\left(\frac{t+1}{k}\right)^4\frac{\epsilon B_2}{(t+1)^2}\\
&\leq \sum_{t=0}^{\infty}\left(\frac{t_0}{k}\right)^4\frac{B_1}{(t+1)^2}+\sum_{t=t_0}^{k_1-1}\frac{B_1}{k^2}+\sum_{t=k_1}^{k-1}\frac{\epsilon B_2}{k^2}\\
&\leq \frac{2t_0^4B_1}{k^4}+\frac{B_1(k_1-t_0)}{k^2}+\frac{\epsilon B_2(k-k_1)}{k^2}\\
&\leq \frac{2\epsilon t_0B_1}{k}+\frac{\epsilon B_1}{k}+\frac{\epsilon B_2}{k}=\frac{\epsilon(B_1(2t_0+1)+B_2)}{k}, \qquad \text{for } k\geq\left\lceil\frac{k_1}{\epsilon}\right\rceil.
\end{aligned}
$$

Thus,

$$
S_3 = o\left(\frac{1}{k}\right). \tag{4.30}
$$

Plugging (4.28)-(4.30) in (4.26) results,

$$
\begin{aligned}
\mathbb{E}[\|\bar{\mathbf{x}}^k-\mathbf{x}^*\|^2] &\leq \frac{(1-\frac{\mu}{L})^{2t_0}t_0^4}{k^4}\|\bar{\mathbf{x}}^0-\mathbf{x}^*\|^2+\frac{4\sigma^2}{3N\mu^2k}+\frac{\sigma^2t_0^4}{NL\mu k^4}+o\left(\frac{1}{k}\right)\\
&= \frac{4\sigma^2}{3N\mu^2k}+o\left(\frac{1}{k}\right).
\end{aligned}
$$

□

4.5 Numerical experiments

To verify our findings and compare different communication strategies in Local SGD, we performed the following numerical experiments, using an Nvidia GTX-1060 GPU and Intel Core i7-7700k processor.

4.5.1 Quadratic function with strong-growth condition

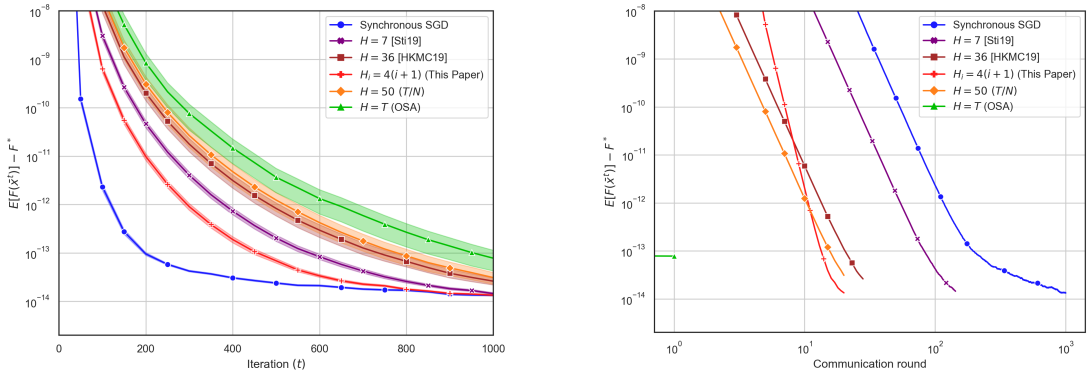
As discussed in Zhang et al. [2016], Dieuleveut and Patel [2019], under uniformly bounded variance, one-shot averaging performs asymptotically as well as mini-batch SGD, at least for quadratic functions. Therefore, to fully capture the importance of the choice of communication times $\mathcal{I}$, we design a *hard* problem, where noise variance is uniform with strong-growth condition, defined in Assumption 8. Let us define,

$$F(\mathbf{x}) = \mathbb{E}_\zeta f(\mathbf{x}, \zeta), \qquad f(\mathbf{x}, \zeta) := \sum_{i=1}^{d} \frac{i}{2} x_i^2 (1 + z_{1,i}) + \mathbf{x}^\top \mathbf{z}_2, \tag{4.31}$$

where $\zeta = (\mathbf{z}_1, \mathbf{z}_2)$ and $\mathbf{z}_1, \mathbf{z}_2 \in \mathbb{R}^d, z_{1,i} \sim \mathcal{N}(0, c_1)$ and $z_{2,i} \sim \mathcal{N}(0, c_2)$, $\forall i \in [d]$, are random variables with normal distributions. We assume at each iteration t, each worker i samples a ζ_i^t and uses $\nabla f(\mathbf{x}, \zeta_i^t)$ as a stochastic estimate of $\nabla F(\mathbf{x})$. It is easy to verify that $F(\mathbf{x})$ is 1-strongly convex and d-smooth, $F^* = 0$ and $\mathbb{E}_\zeta[\|\nabla f(\mathbf{x}, \zeta) - \nabla F(\mathbf{x})\|^2] = c\|\nabla F(\mathbf{x})\|^2 + \sigma^2$, where $c = c_1$ and $\sigma^2 = dc_2$.

We use Local SGD to minimize $F(\mathbf{x})$ using different communication strategies, namely, synchronized SGD where $H = 1$, $H \approx \sqrt{TN}$ [Stich, 2019], $H \approx (TN)^{1/3}$ [Haddadpour et al., 2019], $R = N$ with constant $H \approx T/N$ [Stich and Karimireddy, 2019, Khaled et al., 2020] and finally the communication strategy proposed in this work with $R = N$ and linearly growing H_i local steps. We used $N = 20$ workers, $T = 1000$ iterations, $c_1 = 1.0$ and $c_2 = 10^{-10}$ with $d = 3$ and step-size sequence $\eta_t = 3/(\mu(t + 1))$. To estimate the expected value of errors, we repeated the optimization using each strategy 100 times and reported the average and 1-standard-deviation error bar in Figure 4·2.

We make the following observations from Figure 4·2:



(a) Error over iteration. (b) Error over communication round.

Figure 4·2: Minimizing (4.31) using Local SGD with different communication strategies. Figures (a) and (b) show the error over iteration and communication rounds, respectively.

- Figure 4·2(a) shows that a communication strategy with increasing local steps (proposed in this work), outperforms all the other methods, both in transient and final error performance, specifically the one with the same number of communication rounds evenly spread throughout the whole optimization. This confirms the advantage of more frequent communication at the beginning of the optimization, especially when the ratio of c to σ^2 in the noise with growth condition is large (see the definition in Assumption 8).

- Figure4·2(b) shows that our communication method uses fewer communication rounds, 20 versus 28 [Haddadpour et al., 2019], 143 [Stich, 2019] and 1000 rounds for synchronized SGD.

- OSA appears to perform relatively well despite using only one communication round, though not quite as well as other methods. This shows that the choice of communication is important in this experiment. In other words, it is not true that the success of our communication strategy is merely a byproduct of the experiment design, where any communication strategy, as long as it communicates at least once, will succeed.

4.5.2 Speed-up curves

In this experiment, we minimize a one-dimensional function defined as,

$$F(x) = \begin{cases} \frac{1}{2}x^2, & x \leq 0, \\ x^2, & x > 0, \end{cases} \tag{4.32}$$

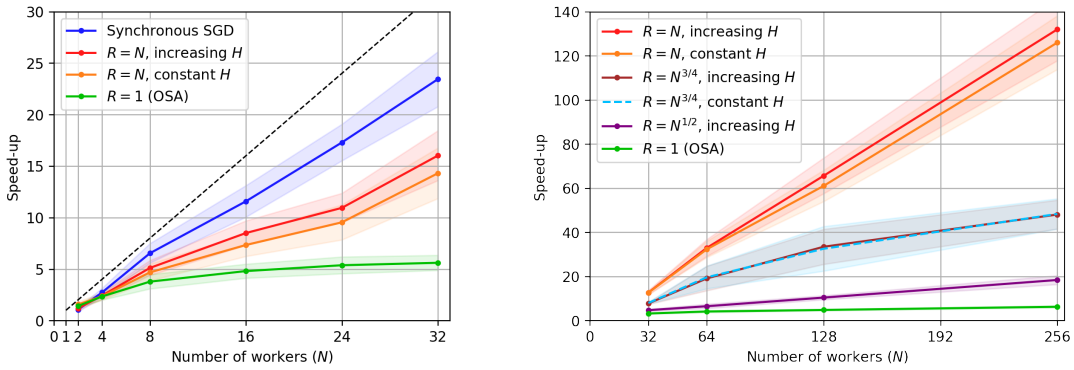
using Local SGD with gradients corrupted by a normal noise $\mathcal{N}(0, \sigma^2)$. We chose this specific cost function since it is not twice continuously differentiable at the minimizer $x^* = 0$ and does not satisfy Assumption 7 required by Theorem 7 for OSA to achieve linear speed-up. The results of this experiment will help us understand whether twice differentiability is a necessary assumption for OSA to obtain a linear speed-up.

The speed-up curve is derived by dividing the *expected* error of a single worker SGD by the *expected* error of each method at the final iterate T, over different number of workers N. Thus in the case where the error decreases linearly in the number of workers, we should expect to see a straight line on the graph.

We plot the speed-up curve for N workers using different communication strategies: synchronized SGD, $R = N$ communication rounds with linearly increasing number of local steps H_i, $R = N$ with constant number of local steps $H \approx T/R$, as well as OSA with only $R = 1$ communication at the end. We use the step-size sequence $\eta_t = \min\{1/L, 2/(\mu(t+1))\}$ with $\mu = 1, L = 2$, and $\sigma = 8$, $T = 1000$.

Our results in Figure 4·3(a) show that Local SGD with $R = N$ (increasing or constant H) achieves linear speed-up in the number of workers, albeit with a worse constant compared to synchronized SGD. However, OSA fails to scale as N increases. This suggests that the condition of twice differentiability (Assumption 7) is necessary for Theorem 7, as this function satisfies all the other assumptions of that theorem.

While our theoretical results provide only an upper bound on R to achieve linear speed-up, this setting gives us a chance to find out if smaller number of communication rounds are enough. Therefore we repeat this experiment for larger number of workers N and $T = 8000$, using $R \approx N^{3/4}$ and $R \approx N^{1/2}$ communication rounds. Our results in Figure 4·3(b) show



(a) (b)

Figure 4·3: Speed-up curves for different communication strategies, over different ranges of N and T. Figure (a) establishes the linear speed-up of local SGD with $R = N$ communication rounds as well as failure of OSA to achieve speed-up even with small number of workers $N \leq 32$ over $T = 1000$ iterations. Figure (b) additionally plots speed-up curves for $R \approx N^{3/4}$ and $R \approx N^{1/2}$ for larger values of $32 \leq N \leq 256$ and $T = 8000$.

that $R = N$ clearly achieves speed-up for larger values of N, as expected and $R = 1$ and $R \approx N^{1/2}$ fail to speed-up. However, $R \approx N^{3/4}$ also struggles to *linearly* speed-up in the number of workers, as the slope of the speed-up curve declines with N increasing. It would be of interest to look into a more granular choice of communication rounds such as $R \approx N^{0.9}$ or even $R \approx N^{0.99}$ but this would require much larger values of N and T and thus more repeated simulations, which is beyond our computational resources, which were already exhausted by generating Figure 2(b).

It is worth mentioning that in both experiments of Figure 4·3(a) and 4·3(b), $R = N$ with increasing H outperforms the one with constant H, even though the noise model used in this experiment is simply uniformly bounded, without strong-growth condition. This further endorses the use of more frequent averaging at the beginning of optimization, when paired with decreasing step-size sequence.

4.5.3 Regularized logistic regression

We consider binary classification and select l_2-regularized logistic regression with its corresponding loss function as the objective function F to be minimized, i.e.,

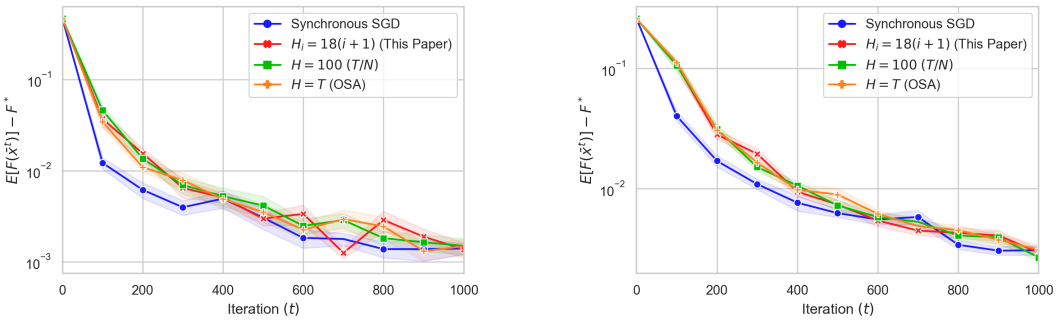
$$F(\mathbf{x}) = \frac{1}{M}\sum_{j=1}^{M}\Big(\ln(1+\exp(\mathbf{x}^\top \mathbf{A}_j)) - 1_{(b_j=1)}\mathbf{x}^\top \mathbf{A}_j\Big) + \frac{\lambda}{2}\|\mathbf{x}\|_2^2, \tag{4.33}$$

where λ is the regularization parameter, $\mathbf{A}_j \in \mathbb{R}^d$ and $b_j \in \{0,1\}$, $j = 1,\ldots,M$ are features (data points) and their corresponding class labels, respectively.

Here we use two large datasets. One, a real dataset from the American College of Surgeons National Surgical Quality Improvement Program (NSQIP) to predict whether a specific patient will be re-admitted within 30 days from discharge after general surgery. This dataset consists of $M = 722{,}101$ data points for training with $d = 231$ features including (i) baseline demographic and healthcare status characteristics, (ii) procedure information and (iii) pre-operative, intra-operative, and post-operative variables. Second, the a9a dataset from LIBSVM [Chang and Lin, 2011]. This dataset consists of $M = 32{,}561$ data points for training with $d = 124$ features.

We perform Local SGD with $N = 10$ workers, $\lambda = 0.05$, step-size sequence $\eta_t = 2/(\mu(t+1))$ ($\beta = 1$), $T = 1000$ iterations and batch size of $b = 1$ with different communication strategies: (i) synchronized SGD with $H = 1$, (ii) a strategy with the time varying communication intervals with $H_i = a(i+1), a \approx 18$ and $R = 10$ communication rounds proposed in this chapter, (iii) a strategy with the same number of communications however with a fixed $H = T/N = 100$, and finally, (iv) one-shot averaging with $H = T$. Each simulation has been repeated 10 times and the average of their performance is reported in Figure 4·4.

It can be seen from Figure 4·4 that all of the communication methods, including OSA, have similar terminal error as synchronized SGD. This further validated our results, especially Theorem 7, since the logistic loss is both twice differentiable and satisfies the PL condition, due to strong convexity of the l_2-regularization. Moreover, we do not notice



(a) NSQIP data set.

(b) a9a data set.

Figure 4·4: Minimizing (4.33) using Local SGD with different communication strategies. Figures (a) and (b) show the error over iteration for NSQIP and a9a datasets, respectively. The shaded areas show the 1-standard deviation error bar.

any significant difference between the performance of the varying and constant local steps, mainly because even a method with only one communication round (OSA) performs just as well.

4.6 Conclusion

In this chapter, we studied the communication complexity of Local SGD and provided an analysis that shows that $R = \Omega(N)$ number of communication rounds, independent of the total number of iterations T, is sufficient to achieve linear speed-up. Moreover, we showed only a single round of averaging is needed provided that the objective is twice differentiable at the optimum point. This assumption appears to be necessary, as our simulations show that not only one-shot averaging but using $N^{1/2}$ or $N^{3/4}$ communications in local SGD fails to deliver linear speed-up on a simple example which is not twice differentiable at the optimum.

Chapter 5

Conclusions

5.1 Summary of the thesis

In this thesis, we tackled communication challenges in distributed methods. We started by looking into average consensus in a directed network with an unreliable communication system. We considered a fully asynchronous push-sum algorithm robust to link failures (RAPS). We proved its convergence while allowing consecutive link failures to grow to infinity, as long as they remain smaller than a logarithmically growing upper bound. We further analyzed RAPS under bounded delays, link failures, and asynchrony and proved its geometrical convergence to average. Moreover, we provided convergence guarantees when the iterates are perturbed.

Then, we turned our attention to the optimization of a sum of local objective functions over a network with a harsh communication setting, that is, a directed network with link failures and delays and asynchrony. We use the results developed for consensus and proposed a Robust Asynchronous Stochastic Gradient Push (RASGP) algorithm and showed that it asymptotically performs as well as its centralized stochastic gradient descent.

Finally, we focused on the problem of speeding up stochastic gradient descent by adding more workers, yet using minimal communication rounds between the processors. This can be achieved by allowing workers to take local steps and then communicate only once in a while. We proposed a new communication strategy and showed that linear speed-up can be achieved in the number of workers N, using only $\mathcal{O}(N)$ communication rounds, independent of the number of total iterations T. Moreover, we showed that under mild additional assumptions, linear speed-up can still be achieved by using only one communication round at the end of optimization, a method called one-shot averaging.

5.2 Future work

Local SGD and federated learning have attracted a lot of attention recently and there is a lot of ongoing research in that area. There are a few potential directions to continue and build on this work.

- Relaxing the I.I.D. assumption for data distribution among workers, allowing heterogeneous data distribution.

- The fixed point of Local SGD (FedAvg) [McMahan et al., 2017] and FedProx [Li et al., 2018] algorithms are not necessarily the global optimum when the data distribution is heterogeneous. Pathak and Wainwright [2020] proposed FedSplit which converges to the optimum while requiring deterministic gradients or Proximal steps with bounded accuracy. Inspired by this work, developing a method using stochastic gradients and optimal fixed-point, would potentially improve upon communication requirements of previous works.

- While we showed empirically that $\mathcal{O}(N)$ communication rounds are needed to achieve linear speed-up, is this tight? Can we also find a theoretical lower bound for the number of communication rounds needed?

- Another comparable algorithm to Local SGD is mini-batch SGD [Woodworth et al., 2020a]. Mini-batch SGD can be seen as the other end of the spectrum from Local SGD, which doesn't allow workers to take any local gradient steps. An interesting area to explore would be a hybrid approach of the two methods: Can we reach better-performing algorithms by having workers take local steps only at some of the iterations, and having a larger batch size for the others?

- Looking at mini-batch SGD, an area to explore would be time-varying mini-batch size and whether we can gain some performance by selecting the mini-batch size in a smarter way, rather than a fixed one.

There are many other directions to continue this work in Federated Learning. See Kairouz et al. [2019] for a comprehensive discussion of open problems in Federated Learning.

Another interesting direction that recently has been more studied, is SGD without replacement (SGDo). In practice, sampling data points without replacement has always shown superior performance compared to sampling with replacement. However, until recently, there was very little theoretical understanding of this phenomenon. Recent paper by Nagaraj et al. [2019] shows $\tilde{\mathcal{O}}(n/T^2)$ convergence rate for *centralized* SGDo over a sum of n smooth and strongly convex functions and T iterations. Moreover, Rajput et al. [2020] show that this bound is tight.

- It would be very interesting to find out how do these new results extend to distributed and decentralized applications or even with Local steps (Local SGDo).

Overall, optimization, centralized or decentralized, has many interesting applications and open directions for future work.

References

Alekh Agarwal and John C Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems*, pages 873–881, 2011.

Mohammad Akbari, Bahman Gharesifard, and Tamás Linder. Distributed online convex optimization on time-varying directed graphs. *IEEE Transactions on Control of Network Systems*, 4(3):417–428, 2017.

Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. In *Advances in Neural Information Processing Systems*, pages 1709–1720, 2017.

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. On the convergence rate of training recurrent neural networks. *arXiv preprint arXiv:1810.12065*, 2018.

Tansu Alpcan and Christian Bauckhage. A distributed machine learning framework. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 2546–2551. IEEE, 2009.

Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. In *Advances in neural information processing systems*, pages 1756–1764, 2015.

Mahmoud Assran and Michael Rabbat. Asynchronous subgradient-push. *arXiv preprint arXiv:1803.08950*, 2018.

Guillermo Barrenetxea, François Ingelrest, Gunnar Schaefer, and Martin Vetterli. Wireless sensor networks for environmental monitoring: The sensorscope experience. In *2008 IEEE International Zurich Seminar on Communications*, pages 98–101. IEEE, 2008.

Amir Beck. *Introduction to nonlinear optimization: Theory, algorithms, and applications with MATLAB*. SIAM, 2014.

Florence Bénézit, Vincent Blondel, Patrick Thiran, John Tsitsiklis, and Martin Vetterli. Weighted gossip: Distributed averaging using non-doubly stochastic matrices. In *IEEE International Symposium on Information Theory (ISIT)*, pages 1753–1757, 2010. doi:10.-1109/ISIT.2010.5513273.

Dimitri P Bertsekas and John N Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

Avleen S Bijral, Anand D Sarwate, and Nathan Srebro. On data dependence in distributed stochastic optimization. *arXiv preprint arXiv:1603.04379*, 2016.

Vincent D Blondel, Julien M Hendrickx, Alex Olshevsky, and John N Tsitsiklis. Convergence in multiagent coordination, consensus, and flocking. In *Proceesings of the 44th IEEE Conference on Decision and Control*, pages 2996–3000. IEEE, 2005. doi:10.1109/-CDC.2005.1582620.

Nicoletta Bof, Ruggero Carli, Giuseppe Notarstefano, Luca Schenato, and Damiano Varagnolo. Newton-raphson consensus under asynchronous and lossy communications for peer-to-peer networks. *arXiv preprint arXiv:1707.09178*, 2017a.

Nicoletta Bof, Ruggero Carli, and Luca Schenato. Average consensus with asynchronous updates and unreliable communication. *IFAC-PapersOnLine*, 50(1):601 – 606, 2017b. ISSN 2405-8963. doi:https://doi.org/10.1016/j.ifacol.2017.08.093. URL `http://www.sciencedirect.com/science/article/pii/S240589631730126X`. 20th IFAC World Congress.

Pierre Brémaud. *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer Science & Business Media, 2013.

Theodora S Brisimi, Ruidi Chen, Theofanie Mela, Alex Olshevsky, Ioannis Ch Paschalidis, and Wei Shi. Federated learning of predictive models from federated electronic health records. *International journal of medical informatics*, 112:59–67, 2018.

Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2012.

Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Tsung-Hui Chang, Mingyi Hong, Wei-Cheng Liao, and Xiangfeng Wang. Asynchronous distributed ADMM for large-scale optimization—part I: algorithm and convergence analysis. *IEEE Transactions on Signal Processing*, 64(12):3118–3130, 2016a.

Tsung-Hui Chang, Wei-Cheng Liao, Mingyi Hong, and Xiangfeng Wang. Asynchronous distributed ADMM for large-scale optimization—part II: Linear convergence analysis and numerical performance. *IEEE Transactions on Signal Processing*, 64(12):3131–3144, 2016b.

Themistoklis Charalambous, Ye Yuan, Tao Yang, Wei Pan, Christoforos N Hadjicostis, and Mikael Johansson. Distributed finite-time average consensus in digraphs in the presence of time delays. *IEEE Transactions on Control of Network Systems*, 2(4):370–381, 2015.

Zachary Charles and Dimitris Papailiopoulos. Stability and generalization of learning algorithms that converge to global optima. In *International Conference on Machine Learning*, pages 745–754. PMLR, 2018.

Jianshu Chen and Ali H Sayed. On the learning behavior of adaptive networks—part II: Performance analysis. *IEEE Transactions on Information Theory*, 61(6):3518–3548, 2015.

Xie Chen, Adam Eversole, Gang Li, Dong Yu, and Frank Seide. Pipelined back-propagation for context-dependent deep neural networks. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

Morris H DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121, 1974.

Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(Jan):165–202, 2012.

Paolo Di Lorenzo and Gesualdo Scutari. Next: In-network nonconvex optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):120–136, 2016.

Aymeric Dieuleveut and Kumar Kshitij Patel. Communication trade-offs for local-sgd with large step size. In *Advances in Neural Information Processing Systems*, pages 13579–13590, 2019.

Alejandro D Dominguez-Garcia and Christoforos N Hadjicostis. Distributed matrix scaling and application to average consensus in directed graphs. *IEEE Transactions on Automatic Control*, 58(3):667–681, 2013.

Alejandro D Domínguez-García and Christoforos N Hadjicostis. Convergence rate of a distributed algorithm for matrix scaling to doubly stochastic form. In *53rd IEEE Conference on Decision and Control*, pages 3240–3245. IEEE, 2014.

Hamid Reza Feyzmahdavian, Arda Aytekin, and Mikael Johansson. An asynchronous mini-batch algorithm for regularized stochastic optimization. *IEEE Transactions on Automatic Control*, 61(12):3740–3754, 2016.

Chunkai Gao, Jorge Cortés, and Francesco Bullo. Notes on averaging over acyclic digraphs and discrete coverage control. *Automatica*, 44(8):2120–2127, 2008.

Federica Garin and Luca Schenato. A survey on distributed estimation and control applications using linear consensus algorithms. In *Networked control systems*, pages 75–107. Springer, 2010.

Balázs Gerencsér and Julien M. Hendrickx. Push sum with transmission failures. *CoRR*, abs/1504.08193, 2015. URL `http://arxiv.org/abs/1504.08193`.

Bahman Gharesifard and Jorge Cortés. Distributed strategies for generating weight-balanced and doubly stochastic digraphs. *European Journal of Control*, 18(6):539–557, 2012.

Antoine Godichon-Baggioni and Sofiane Saadane. On the rates of convergence of parallelized averaged stochastic gradient algorithms. *Statistics*, 54(3):618–635, 2020.

Demjan Grubic, Leo K Tam, Dan Alistarh, and Ce Zhang. Synchronous multi-gpu deep learning with low-precision communication: An experimental study. In *Proceedings of the 21st International Conference on Extending Database Technology*, pages 145–156. OpenProceedings, 2018.

Farzin Haddadpour, Mohammad Mahdi Kamani, Mehrdad Mahdavi, and Viveck Cadambe. Local sgd with periodic averaging: Tighter analysis and adaptive synchronization. In *Advances in Neural Information Processing Systems*, pages 11080–11092, 2019.

Christoforos N Hadjicostis and Themistoklis Charalambous. Average consensus in the presence of delays and dynamically changing directed graph topologies. *arXiv preprint arXiv:1210.4778*, 2012.

Christoforos N Hadjicostis and Themistoklis Charalambous. Average consensus in the presence of delays in directed graph topologies. *IEEE Transactions on Automatic Control*, 59(3):763–768, 2014.

Christoforos N Hadjicostis, Nitin H Vaidya, and Alejandro D Domínguez-García. Robust distributed average consensus via exchange of running sums. *IEEE Transactions on Automatic Control*, 61(6):1492–1507, 2016.

Christoforos N Hadjicostis, Alejandro D Domínguez-García, Themistokis Charalambous, et al. Distributed averaging and balancing in network systems: with applications to coordination and control. *Foundations and Trends® in Systems and Control*, 5(2-3): 99–292, 2018.

Shibo He, Dong-Hoon Shin, Junshan Zhang, Jiming Chen, and Youxian Sun. Full-view area coverage in camera sensor networks: Dimension reduction and near-optimal solutions. *IEEE Transactions on Vehicular Technology*, 65(9):7448–7461, 2015.

Hadrien Hendrikx, Francis Bach, and Laurent Massoulié. An accelerated decentralized stochastic proximal algorithm for finite sums. In *Advances in Neural Information Processing Systems*, pages 952–962, 2019.

Mingyi Hong. A distributed, asynchronous and incremental algorithm for nonconvex optimization: An ADMM approach. *IEEE Transactions on Control of Network Systems*, 5 (3):935–945, 2017. doi:10.1109/TCNS.2017.2657460.

Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Keith Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *arXiv preprint arXiv:1912.04977*, 2019.

Hamed Karimi, Julie Nutini, and Mark Schmidt. Linear convergence of gradient and proximal-gradient methods under the polyak-łojasiewicz condition. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 795–811. Springer, 2016.

David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, pages 482–491. IEEE, 2003.

A Khaled, K Mishchenko, and P Richtárik. Tighter theory for local sgd on identical and heterogeneous data. In *The 23rd International Conference on Artificial Intelligence and Statistics (AISTATS 2020)*, 2020.

Anastasiia Koloskova, Sebastian Urban Stich, and Martin Jaggi. Decentralized stochastic optimization and gossip algorithms with compressed communication. *Proceedings of Machine Learning Research*, 97(CONF), 2019.

Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian U Stich. A unified theory of decentralized sgd with changing topology and local updates. *arXiv preprint arXiv:2003.10422*, 2020.

Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

Guanghui Lan, Soomin Lee, and Yi Zhou. Communication-efficient algorithms for decentralized and stochastic optimization. *Mathematical Programming*, pages 180, 238–284, 2020. doi:10.1007/s10107-018-1355-4.

Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 14)*, pages 583–598, 2014.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *arXiv preprint arXiv:1812.-06127*, 2018.

Xiangru Lian, Yijun Huang, Yuncheng Li, and Ji Liu. Asynchronous parallel stochastic gradient for nonconvex optimization. In *Advances in Neural Information Processing Systems*, pages 2737–2745, 2015.

Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, and Ji Liu. Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 5330–5340, 2017.

Xiangru Lian, Wei Zhang, Ce Zhang, and Ji Liu. Asynchronous decentralized parallel stochastic gradient descent. In *International Conference on Machine Learning (ICML)*, pages 3043–3052, 2018.

Tao Lin, Sebastian U Stich, Kumar Kshitij Patel, and Martin Jaggi. Don't use large mini-batches, use local sgd. *arXiv preprint arXiv:1808.07217*, 2018a.

Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. In *International Conference on Learning Representations*, 2018b. URL `https://openreview.net/forum?id=SkhQHMW0W`.

Ilan Lobel and Asuman Ozdaglar. Distributed subgradient methods for convex optimization over random networks. *IEEE Transactions on Automatic Control*, 56(6):1291–1306, 2010.

Jan Lorenz. Convergence to consensus in multiagent systems and the lengths of intercommunication intervals. *arXiv preprint arXiv:1101.2926*, 2011.

Liam Madden, Emiliano Dall'Anese, and Stephen Becker. High probability convergence and uniform stability bounds for nonconvex stochastic gradient descent. *arXiv preprint arXiv:2006.05610*, 2020.

Fatemeh Mansoori and Ermin Wei. Superlinearly convergent asynchronous distributed network newton method. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2874–2879. IEEE, 2017.

Ryan Mcdonald, Mehryar Mohri, Nathan Silberman, Dan Walker, and Gideon S Mann. Efficient large-scale distributed training of conditional maximum entropy models. In *Advances in neural information processing systems*, pages 1231–1239, 2009.

Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. In *Human language technologies: The 2010 annual conference of the North American chapter of the association for computational linguistics*, pages 456–464. Association for Computational Linguistics, 2010.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Statistics*, pages 1273–1282, 2017.

Gemma Morral, Pascal Bianchi, Gersende Fort, and Jérémie Jakubowicz. Distributed stochastic approximation: The price of non-double stochasticity. In *Conference Record of the Forty Sixth Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 1473–1477. IEEE, 2012.

Gemma Morral, Pascal Bianchi, and Gersende Fort. Success and failure of adaptation-diffusion algorithms with decaying step size in multiagent networks. *IEEE Transactions on Signal Processing*, 65(11):2798–2813, 2017.

Dheeraj Nagaraj, Prateek Jain, and Praneeth Netrapalli. Sgd without replacement: Sharper rates for general smooth convex functions. In *International Conference on Machine Learning*, pages 4703–4711. PMLR, 2019.

Angelia Nedić. Asynchronous broadcast-based convex optimization over a network. *IEEE Transactions on Automatic Control*, 56(6):1337–1351, 2011.

Angelia Nedić and Alex Olshevsky. Distributed optimization over time-varying directed graphs. *IEEE Transactions on Automatic Control*, 60(3):601–615, 2015.

Angelia Nedić and Alex Olshevsky. Stochastic gradient-push for strongly convex functions on time-varying directed graphs. *IEEE Transactions on Automatic Control*, 61(12): 3936–3947, 2016.

Angelia Nedić and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.

Angelia Nedić, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27(4): 2597–2633, 2017.

Angelia Nedić, Alex Olshevsky, and Michael G Rabbat. Network topology and communication-computation tradeoffs in decentralized optimization. *Proceedings of the IEEE*, 106 (5):953–976, 2018.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Alex Olshevsky. Efficient information aggregation strategies for distributed control and signal processing. *arXiv preprint arXiv:1009.6036*, 2010.

Alex Olshevsky. Linear time average consensus on fixed graphs and implications for decentralized optimization and multi-agent control. *arXiv preprint arXiv:1411.4186*, 2014.

Alex Olshevsky. Linear time average consensus and distributed optimization on fixed graphs. *SIAM Journal on Control and Optimization*, 55(6):3990–4014, 2017.

Alex Olshevsky, Ioannis Ch Paschalidis, and Artin Spiridonoff. Fully asynchronous push-sum with growing intercommunication intervals. *American Control Conference*, pages 591–596, 2018.

Boris N Oreshkin, Mark J Coates, and Michael G Rabbat. Optimization and analysis of distributed averaging with short node memory. *IEEE Transactions on Signal Processing*, 58(5):2850–2865, 2010.

Reese Pathak and Martin J Wainwright. Fedsplit: An algorithmic framework for fast federated optimization. *arXiv preprint arXiv:2005.05238*, 2020.

Zhouhua Peng, Jun Wang, and Dan Wang. Distributed maneuvering of autonomous surface vehicles based on neurodynamic optimization and fuzzy approximation. *IEEE Transactions on Control Systems Technology*, 26(3):1083–1090, 2017.

Shi Pu and Alfredo Garcia. A flocking-based approach for distributed stochastic optimization. *Operations Research*, 66(1):267–281, 2017.

Shi Pu and Angelia Nedić. A distributed stochastic gradient tracking method. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 963–968. IEEE, 2018.

Guannan Qu and Na Li. Harnessing smoothness to accelerate distributed optimization. *IEEE Transactions on Control of Network Systems*, 5(3):1245–1260, 2018. doi:10.1109/-TCNS.2017.2698261.

Guannan Qu and Na Li. Accelerated distributed Nesterov gradient descent. *IEEE Transactions on Automatic Control*, 65(6):2566–2581, 2019. doi:10.1109/TAC.2019.2937496.

Shashank Rajput, Anant Gupta, and Dimitris Papailiopoulos. Closing the convergence gap of sgd without replacement. In *International Conference on Machine Learning*, pages 7964–7973. PMLR, 2020.

Alexander Rakhlin, Ohad Shamir, Karthik Sridharan, et al. Making gradient descent optimal for strongly convex stochastic optimization. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, volume 12, pages 1571–1578. Citeseer, 2012.

S Sundhar Ram, Angelia Nedić, and Venugopal V Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications*, 147(3):516–545, 2010.

Benjamin Recht, Christopher Re, Stephen Wright, and Feng Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems*, pages 693–701, 2011.

Jason DM Rennie and Nathan Srebro. Loss functions for preference levels: Regression with discrete ordered labels. In *Proceedings of the IJCAI multidisciplinary workshop on advances in preference handling*, pages 180–186. Kluwer Norwell, MA, 2005.

Pouya Rezaeinia, Bahman Gharesifard, Tamas Linder, and Behrouz Touri. Push-sum on random graphs. *arXiv preprint arXiv:1708.00915*, 2017.

Jonathan D Rosenblatt and Boaz Nadler. On the optimality of averaging in distributed statistical learning. *Information and Inference: A Journal of the IMA*, 5(4):379–404, 2016.

Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)-Volume 70*, pages 3027–3036. JMLR. org, 2017.

Mark Schmidt and Nicolas Le Roux. Fast convergence of stochastic gradient descent under a strong growth condition. *arXiv preprint arXiv:1308.6370*, 2013.

Eugene Seneta. *Non-negative matrices and Markov chains*. Springer Science & Business Media, 2006.

Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow. *arXiv preprint arXiv:1802.05799*, 2018.

Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.

Benjamin Sirb and Xiaojing Ye. Consensus optimization with delayed and stochastic gradients on decentralized networks. In *2016 IEEE International Conference on Big Data (Big Data)*, pages 76–85. IEEE, 2016.

Artin Spiridonoff, Alex Olshevsky, and Ioannis Ch Paschalidis. Robust asynchronous stochastic gradient-push: asymptotically optimal and network-independent performance for strongly convex functions. *Journal of Machine Learning Research*, 2020.

Artin Spiridonoff, Alex Olshevsky, and Ioannis Ch Paschalidis. Communication-efficient sgd: From local sgd to one-shot averaging. *arXiv preprint arXiv:2106.04759*, 2021.

Kunal Srivastava and Angelia Nedić. Distributed asynchronous constrained stochastic optimization. *IEEE Journal of Selected Topics in Signal Processing*, 5(4):772–790, 2011.

Sebastian U. Stich. Local SGD converges fast and communicates little. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=S1g2JnRcFX`.

Sebastian U Stich and Sai Praneeth Karimireddy. The error-feedback framework: Better rates for sgd with delayed gradients and compressed communication. *arXiv:1909.05350*, 2019.

Sebastian U Stich, Jean-Baptiste Cordonnier, and Martin Jaggi. Sparsified sgd with memory. In *Advances in Neural Information Processing Systems*, pages 4447–4458, 2018.

Lili Su and Nitin H Vaidya. Fault-tolerant multi-agent optimization: optimal iterative distributed algorithms. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, pages 425–434. ACM, 2016a.

Lili Su and Nitin H Vaidya. Non-bayesian learning in the presence of byzantine agents. In *International Symposium on Distributed Computing*, pages 414–427. Springer, 2016b.

Lili Su and Nitin H. Vaidya. Reaching approximate byzantine consensus with multi-hop communication. *Information and Computation*, 255:352 – 368, 2017. ISSN 0890-5401. doi:https://doi.org/10.1016/j.ic.2016.12.003. URL `http://www.sciencedirect.com/science/article/pii/S0890540116301262`. SSS 2015.

Ying Sun, Gesualdo Scutari, and Daniel Palomar. Distributed nonconvex multiagent optimization over time-varying networks. In *50th Asilomar Conference on Signals, Systems and Computers*, pages 788–794. IEEE, 2016.

Zhenheng Tang, Shaohuai Shi, Xiaowen Chu, Wei Wang, and Bo Li. Communication-efficient distributed deep learning: A comprehensive survey. *arXiv:2003.06307*, 2020.

Ye Tian, Ying Sun, and Gesualdo Scutari. Asy-sonata: Achieving linear convergence in distributed asynchronous multiagent optimization. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 543–551. IEEE, 2018.

Konstantinos I Tsianos, Sean Lawlor, and Michael G Rabbat. Consensus-based distributed optimization: Practical issues and applications in large-scale machine learning. In *50th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1543–1550. IEEE, 2012a. doi:10.1109/Allerton.2012.6483403.

Konstantinos I Tsianos, Sean Lawlor, and Michael G Rabbat. Push-sum distributed dual averaging for convex optimization. In *IEEE 51st Annual Conference on Decision and Control (CDC)*, pages 5453–5458. IEEE, 2012b. doi:10.1109/CDC.2012.6426375.

John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control*, 31(9):803–812, 1986.

John Nikolas Tsitsiklis. Problems in decentralized decision making and computation. Technical report, Massachusetts Inst of Tech Cambridge Lab for Information and Decision Systems, 1984. URL `https://www.mit.edu/~jnt/Papers/PhD-84-jnt.pdf`.

Damiano Varagnolo, Filippo Zanella, Angelo Cenedese, Gianluigi Pillonetto, and Luca Schenato. Newton-raphson consensus for distributed convex optimization. *IEEE Transactions on Automatic Control*, 61(4):994–1009, 2016.

Jianyu Wang and Gauri Joshi. Adaptive communication strategies to achieve the best error-runtime trade-off in local-update sgd. *Proceedings of the 2nd SysML Conference*, 2018a. URL `https://mlsys.org/Conferences/2019/doc/2019/124.pdf`.

Jianyu Wang and Gauri Joshi. Cooperative sgd: A unified framework for the design and analysis of communication-efficient sgd algorithms. *arXiv preprint arXiv:1808.07576*, 2018b.

Jianyu Wang, Anit Kumar Sahu, Zhouyi Yang, Gauri Joshi, and Soummya Kar. Matcha: Speeding up decentralized sgd via matching decomposition sampling. *arXiv preprint arXiv:1905.09435*, 2019.

Jianqiao Wangni, Jialei Wang, Ji Liu, and Tong Zhang. Gradient sparsification for communication-efficient distributed optimization. In *Advances in Neural Information Processing Systems*, pages 1299–1309, 2018.

Blake Woodworth, Kumar Kshitij Patel, and Nathan Srebro. Minibatch vs local sgd for heterogeneous distributed learning. *arXiv preprint arXiv:2006.04735*, 2020a.

Blake Woodworth, Kumar Kshitij Patel, Sebastian U Stich, Zhen Dai, Brian Bullins, H Brendan McMahan, Ohad Shamir, and Nathan Srebro. Is local sgd better than minibatch sgd? *arXiv preprint arXiv:2002.07839*, 2020b.

Tianyu Wu, Kun Yuan, Qing Ling, Wotao Yin, and Ali H Sayed. Decentralized consensus optimization with asynchrony and delays. *IEEE Transactions on Signal and Information Processing over Networks*, 4(2):293–307, 2018.

Chenguang Xi and Usman A Khan. Dextra: A fast algorithm for optimization over directed graphs. *IEEE Transactions on Automatic Control*, 62(10):4980–4993, 2017a.

Chenguang Xi and Usman A Khan. Distributed subgradient projection algorithm over directed graphs. *IEEE Transactions on Automatic Control*, 62(8):3986–3992, 2017b.

Chenguang Xi, Ran Xin, and Usman A Khan. Add-opt: Accelerated distributed directed optimization. *IEEE Transactions on Automatic Control*, 63(5):1329–1339, 2018.

Jinming Xu, Shanying Zhu, Yeng Chai Soh, and Lihua Xie. Augmented distributed gradient methods for multi-agent optimization under uncoordinated constant stepsizes. In *2015 54th IEEE Conference on Decision and Control (CDC)*, pages 2055–2060. IEEE, 2015.

Omry Yadan, Keith Adams, Yaniv Taigman, and Marc'Aurelio Ranzato. Multi-gpu training of convnets. *arXiv preprint arXiv:1312.5853*, 2013.

Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5693–5700, 2019.

Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.

Jian Zhang, Christopher De Sa, Ioannis Mitliagkas, and Christopher Ré. Parallel sgd: When does averaging help? *arXiv preprint arXiv:1606.07365*, 2016.

Shanshan Zhang, Ce Zhang, Zhao You, Rong Zheng, and Bo Xu. Asynchronous stochastic gradient descent for dnn training. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6660–6663. IEEE, 2013a.

Sixin Zhang, Anna E Choromanska, and Yann LeCun. Deep learning with elastic averaging sgd. In *Advances in neural information processing systems*, pages 685–693, 2015.

Yuchen Zhang, John Duchi, Michael I Jordan, and Martin J Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems*, pages 2328–2336, 2013b.

Yuchen Zhang, John C Duchi, and Martin J Wainwright. Communication-efficient algorithms for statistical optimization. *Journal of Machine Learning Research*, 14(1): 3321–3363, 2013c.

Fan Zhou and Guojing Cong. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3219–3227. AAAI Press, 2018.

Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. In *Advances in neural information processing systems*, pages 2595–2603, 2010.

CURRICULUM VITAE